

Compute-in-Memory Implementation of State Space Models for Event Sequence Processing

Corresponding Author: Professor Wei Lu

This file contains all reviewer reports in order by version, followed by all author rebuttals in order by version.

Version 0:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

The manuscript entitled 'Compute-in-Memory Implementation of State Space Models for Event Sequence Processing' presents a SSM & CIM co-design method to implement efficient asynchronous processing for event sequences. This work is noteworthy in replacing the numeric iteration in the SSM with the short term memory effect of WOx devices, while using RRAM for VMM operations. The authors claimed that the proposed method outperforms existing SNN-based methods with respect to benchmark performance and power. However, the manuscript exhibits several weaknesses that require further improvement, as detailed below.

- 1) The authors present their method as a real-time, event-driven hardware system. Therefore, it is expected to demonstrate a complete system with sufficient design details and measurement results. Critical parts should include the WOx-based SSM block, RRAM-based embedding layer and VMM modules, as well as how they work (e.g., how to implement line 338-341 physically?). Fig. 1 and 3 show several highly abstracted figures without details, making a big gap between the proposed concept and the hardware implementation. Moreover, the CIM chip of Fig. 3(A) is the only hardware image I can find in this paper, and again, without any details. It is suggested to provide a more thorough documentation of the hardware implementation to allow readers to follow.
- 2) It is not clear whether the results in Table 1 and 2 were obtained through simulation or hardware measurement. Please clarify in both caption and main text. It is hard to "imagine" the deployment of millions of parameters onto the proposed system without knowing the implementation details. If it was done by simulation, please provide full details about the simulation model, e.g. how was the noise modelled?
- 3) Following the last comment, please clearly present the data source throughout the manuscript. It is known the figure 3(C), (D) and S2 were experimentally measured, while fig 4(B) was simulated by TSMC PDK. What about other data, e.g. Fig 2? For the above reasons, it is hard to tell whether the results are technically sound.
- 4) The authors are mixing multiple concepts, which may hinder reader comprehension. The term Reservoir Node should originate from reservoir computing. The authors previously published reservoir computing works with the same device WOx and the same decay property (ref. 58). As the decay factor λ in eq. 2 and 9 are determined by the WOx devices, it is fixed and less likely to be trained. Therefore the definition appears to be in line with reservoir computing. In fact, reservoir computing can be considered a SSM as the reservoir layer produces a state space. The authors should clarify the necessity of this new terminology and discuss how their SSM-based approach different from reservoir computing frameworks.
- 5) The training process could be described in more detail. Please provide more details about the implementation of BPTT and how it handles the noisy dynamical behavior of WOx and RRAM? Are they differentiable? Also, it would be helpful to show the loss curve for the tasks.
- 6) The authors claimed fully asynchronous processing. But, could the authors clarify how the claimed asynchronicity is maintained given that the ADC/DAC likely require clocks? The extent of asynchronicity needs to be clarified.
- 7) In Line 374, the authors didn't explain how to measure the power of RN devices and CIM chip. This is an important part that would benefit from more details.
- 8) Both figures and text are poorly organized. The main problems are:
 - (i) fig 1 (B) and (C) are somewhat unclear. Why the embedding vector is pointed to a straight line? It is not understood what exactly is the embedding. Both captions mention 'internal structure'. The current figures may illustrate some functionalities but the structure is not clearly depicted.
 - (ii) Insufficient labeling in figures obscures essential experimental/design details. The authors should label all the terms. For instance, what are Δt_1 and Δt_2 in Fig 1(c)? What does every arrow means in fig 4(A)? In fig. 4(B), the power distribution percentages quoted in the text (lines 387-389) do not match those shown in Figure 4(B). And What is LUT? It has

not been mentioned throughout the text. In table 2, the acc. for DVS128 is 63.5%, that is, different from the 62.8% in line 280. Please double check the paper.

(iii) It is suggested to totally reorganize the Results and Methods section. The majority of line 121-145 looks like a figure caption. And there are too many method descriptions in the Result section. Please first clear up what information should be in the Results, Methods, figure caption and SI, and then rewrite accordingly.

(iv) The error of Fig 3(A) should be quantified using e.g. NRMSE.

(v) The authors put the comparison table after fig 1. It is suggested to restructure the paper to follow a logical progression: problem -> algorithm -> hardware -> validation -> comparison.

Minor:

Line 142: simply -> simplified

Line 326: double check the ref. 30. RN wasn't found here.

Line 374: system performance or power consumption?

Line 40: Transformer block wasn't found in ref. 11,12 and 13.

Reviewer #2

(Remarks to the Author)

This paper presents a methodology for implementing SSMs on CIM hardware, and holds significant potential by crossing over two emerging fields. In my view, this paper can enable significant impact. Some minor notes that could enhance the quality and technical accuracy of the manuscript are provided below. Note there are several references used to back up several points in this review. I do not expect the authors to include these citations.

Introduction

- The authors state that the use of surrogate gradients is what compromises model accuracy. While this is broadly the accepted view and commonly cited reason, I'd contend that surrogate gradients are generally quite robust (especially in the era of QAT and extreme low-precision models that have been shown to scale up). I expect the main limiter is binarization of activations. This drives excessive information compression. This constraint is highlighted in Zhu et al.'s work in SpikeGPT (TMLR, 2024).

- The paragraph that highlights the mismatch for convolutions and transformers for CIM makes a lot of sense. Buchel & Sebastian et al.'s work in "Efficient scaling of large language models with mixture of experts and 3D analog in-memory computing", Nat Comp Sci, 2025 is a good reference for this, where all Transformer/self-attention operations are delegated to digital circuitry. However, an additional challenge is that applying CIM to dense-layers alone does not address the primary bottleneck of Transformer ops for long sequence length. I.e., Transformers will dominate the computational cost relative to dense layers. This paragraph (or the start of the following paragraph) could be made more compelling by more clearly stating that SSMs overcome the memory-bottleneck of self-attention/KV-caching, making them a natural fit for CIM architectures.

- There is some tightly related work where SSMs are deployed on neuromorphic/event-based hardware by Abreu et al. (Neuromorphic Principles for Efficient Large Language Models on Intel Loihi 2, ICLR SCOPE Workshop, 2025) & Zhu et al. (Scalable MatMul-Free Language Modeling on Neuromorphic Hardware, arXiv 2025). Neither uses CIM, and instead relies on digital, async accelerators.

- I would encourage citing the original SSM+DL work by Voelker, Kajic & Elias Smith, "Legendre memory units: Continuous-time representation in recurrent neural networks", neurips, 2018, where referring to "grounded in control theory".

- The use of real-valued coefficients in the transition matrix is acceptable, and the use of non-dynamic/input-dependent transition matrices for the update is a fair fit for CIM, though the dynamic transition matrix is a big part of what boosts performance in recurrent architectures. I think it is fair to state frankly that this is a limitation, although, if it was delegated to digital computation, it's a tiny fraction of the overall compute.

Results

- The graph in Figure 1(B) is unclear to me (the square).

- The discussion on nonlinearities could use the phrase "positional nonlinearity" which 1) preserves universal function approximation, 2) without breaking the ability to represent training as 1D-conv.

- In model training, it's said that BPTT is used. Training an SSM is indeed functionally equivalent to BPTT, though some folk take issue with it as BPTT is defined as having linear time complexity while training SSMs follows log-complexity with time. I don't think there is concrete guidance here - I have no fixation on this if you prefer not to change this.

- It is unclear to me where the asynchronicity in the SSM comes from. What is sparsifying the activations?

- While not essential, an ablation study to identify what features of the RRAM/ADC non-idealities impact performance.

- There is limited information on how sparsity provides an advantage in this particular set up.

Reviewer #3

(Remarks to the Author)

This manuscript presents an algorithm-hardware co-design approach for efficient, real-time, and asynchronous processing using state space models (SSMs). The key contributions can be summarized as follows: (1) a re-parameterized 6-block SSM model in which the decay parameter (λ) is shared within each block, thereby reducing hardware complexity while improving accuracy; (2) a memristor-based compute-in-memory (CIM) architecture to accelerate input vector-matrix multiplications (VMMs); and (3) an exploitation of the intrinsic short-term conductance decay behavior of memristors to perform hidden state (h) updates. However, several parts of the manuscript lack clarity and require further elaboration and justification.

1. Recent References on RRAM CIM

The reference list regarding RRAM CIM could be strengthened by including recent works published in Nature and Nature Electronics. Incorporating these high-impact studies would better position the manuscript within the current state of the art and highlight its relevance to ongoing developments.

2. Crossbar vs. 1T1R Structures

The manuscript should address the impact of sneak current in crossbar arrays and provide a discussion comparing the advantages and trade-offs of crossbar versus 1T1R structures. Such an analysis would help readers better appreciate the design choices and their implications for scalability and reliability.

3. Figure 1(A) & (B): Clarity of Asynchronous Event Representation

The presentation of asynchronous events through the channel-wise embedding layer is unclear. For instance, the y-axis values are not specified, and it is difficult to determine whether the spikes in (A) correspond directly to the square wave in (B). Providing clarification or additional explanation would greatly improve readability.

4. Synchronous vs. Asynchronous Processing

The manuscript emphasizes the importance of asynchronous processing, but the distinction from synchronous processing is not sufficiently detailed. The authors are encouraged to elaborate on the differences, including their respective advantages and disadvantages. For example, in Table 2, Peng et al. [50] report comparable performance despite not being asynchronous. Enhancing this comparison table to highlight the significance of the proposed approach would strengthen the manuscript.

5. Figure 2(A–D): Caption and Interpretation

The caption states “Analysis of model architecture and decay parameter (λ) choices on classification accuracy,” yet the figure appears to show the number of blocks versus accuracy. Does the caption misstate the content, or is there a specific relationship between λ and the number of blocks? Furthermore, the authors claim that the proposed model requires only six blocks, whereas counterparts such as ResNet require more than 30 layers. Given the distinction between the terms “block” and “layer,” is it valid to conclude that the proposed model achieves lower latency? A direct comparison in terms of latency or other performance metrics would make this claim more convincing.

6. RRAM Mapping Strategy

The authors map positive and negative weights onto separate RRAM arrays, while prior work often employs 2's complement mapping, requiring only a single array. What motivated the use of separate arrays in this work? Discussion of the resulting area overhead and associated trade-offs would add valuable context.

7. Fixed Decay Parameter (λ) and Accuracy

Line 276 states that inference accuracy improves when fixing the decay parameter (λ). The authors should provide a more detailed mathematical derivation or theoretical explanation to clarify why and how fixing λ enhances accuracy.

8. Flexibility Limitations of Memristor Conductance Decay

Using memristor conductance decay as RN is an intriguing approach. However, because the decay rate is largely determined by oxide thickness during fabrication, flexibility may be limited for supporting different SSM models or enabling weight updates. The authors should elaborate on potential solutions to overcome this limitation, particularly given the importance of reprogramming capabilities at the edge.

9. Voltage Levels and Endurance in Conductance Updates

In Equation (9), the horizontal line voltages of the input projection memristor array are used as input spikes $x(t+\Delta t)$ to update conductance. The authors should clarify the voltage levels applied in this process. Moreover, since each conductance update effectively involves multiple write operations, a discussion of endurance concerns under repeated updates would be valuable.

Version 1:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

I thank the authors for having carefully taken my comments into account. The authors made substantial improvements on the manuscript. The claimed points are now endorsed by enough details and results. My only two minor comments are: 1) please avoid using some strange wordings like 'software-level'. Do you mean simulation? If so please simply use 'simulation'. 2) Now I understand that the main difference between the proposed method and reservoir computing is that the decay factor λ of the devices were also trained, providing extra flexibilities for the system to better adapt to the tasks. My only concern is that was the λ trained offline and then somehow configured the devices to the values of target λ ? Please clarify. If so, how to overcome the device variabilities and sim2real problems? Apart from these minor comments, I can recommend publication in Nature communications.

Reviewer #2

(Remarks to the Author)

The authors have fully addressed all of my concerns. I expect this paper will help advance the field.

Reviewer #3

(Remarks to the Author)

The previous comments have been addressed by the authors.

1. However, I am wondering why the authors claim that the transformer is not compatible with CIM, given that numerous CIM studies have been proposed for transformer-based architectures.

2. Additionally, regarding the decay parameter (λ), the authors mention that a fixed value is used. Could the authors discuss whether this parameter may vary due to temperature fluctuations or temporal decay, and how such variations might affect the application performance or computational accuracy?

Version 2:

Reviewer comments:

Reviewer #1

(Remarks to the Author)

Thank you for your efforts. I have no further comments and recommend publication.

Reviewer #2

(Remarks to the Author)

The authors have addressed all of my previous points. The manuscript is in strong shape and should have a considerable degree of impact.

Reviewer #3

(Remarks to the Author)

I thank the author for addressing my comments. I have no further questions.

Open Access This Peer Review File is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license, and indicate if changes were made.

In cases where reviewers are anonymous, credit should be given to 'Anonymous Referee' and the source.

The images or other third party material in this Peer Review File are included in the article's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

To view a copy of this license, visit <https://creativecommons.org/licenses/by/4.0/>

Response Letter

Title:

Compute-in-Memory Implementation of State Space Models for Event Sequence Processing

We thank all reviewers for their valuable and constructive comments on this paper. We have revised the manuscript based on the reviewers' valuable suggestions and comments. The detailed point-by-point response can be found below.

Reviewer #1:

Comments to the Author:

1. The authors present their method as a real-time, event-driven hardware system. Therefore, it is expected to demonstrate a complete system with sufficient design details and measurement results. Critical parts should include the *WO*_x-based SSM block, RRAM-based embedding layer and VMM modules, as well as how they work (e.g., how to implement line 338-341 physically?). Fig. 1 and 3 show several highly abstracted figures without details, making a big gap between the proposed concept and the hardware implementation. Moreover, the CIM chip of Fig. 3(A) is the only hardware image I can find in this paper, and again, without any details. It is suggested to provide a more thorough documentation of the hardware implementation to allow readers to follow.

Response:

We thank the reviewer for this valuable comment. Following the reviewer's suggestions, we have substantially expanded the "Hardware Implementation" section, revised Figure 1 and Figure 2 (formerly Figure 3) and provided a more thorough description of the hardware design and measurement results.

The main changes are as follows:

We revised Figure 1 so that it focuses on the model architecture with more details, and revised Figure 2 (previous Figure 3) so that it focuses on hardware implementation with more details:

Revised Figure 1:

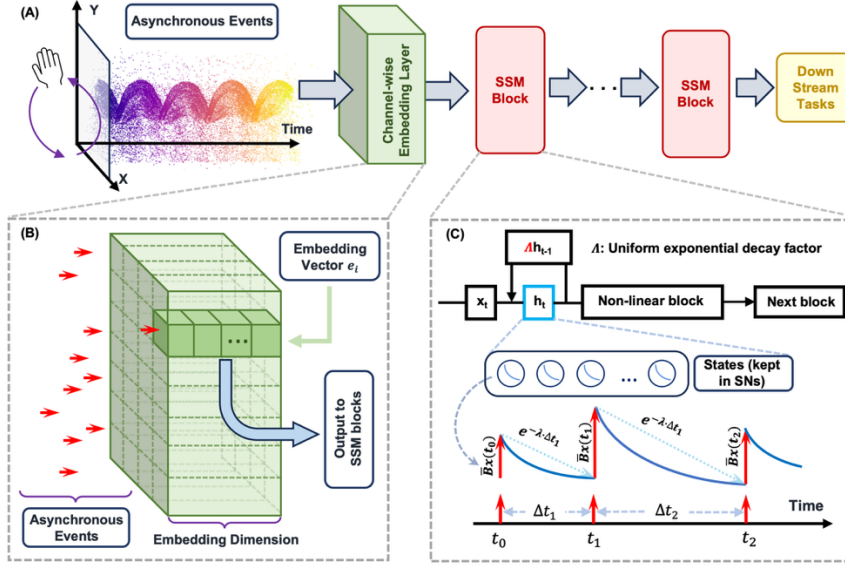


Figure 1. Model architecture for the proposed event-driven SSM. (A) High-level network architecture, including a channel-wise embedding layer that maps incoming events to a high-dimensional latent state space and stacked SSM blocks that asynchronously process information in the state space; (B) Functional diagram of the channel-wise embedding layer. An incoming spike activates the embedding layer which produces a D -dimensional vector depending on the event's channel index. (C) Operational schematic of a single SSM block (upper). The state of the SSM is represented by those of individual state nodes (SNs) (middle) in the proposed implementation. The state in an individual SN decays exponentially following a trained decay time constant. When an input x arrives, it is projected onto the state space through $\bar{B}x$ and is added to the current state. The updated state h is then read out, passed through some non-linear functions and sent to the next block. Note actions (i.e. computations) are only performed when an event arrives, allowing asynchronous operations.

To improve the clarity of our hardware implementation, we have revised the schematic in Figure 2(A) to clearly partition the different functional units in an SSM block in the hardware setup, and color match these functional units with those in the algorithm. In the revised text and figure caption, we also clearly stated how the functional units, e.g. the VMM unit (colored in blue) and the state nodes (colored in red) are implemented using the CIM chip and the WO_x devices, as shown in Figure 2(B) and Figure 2(C-D) respectively.

We also included an SEM image of the fabricated WO_x devices that implemented the state update functions in the inset of Figure 2(D), as well as the distribution of the output from the VMM operations due to device variations as an inset to Figure 2(B). The new Figure 2 is shown below:

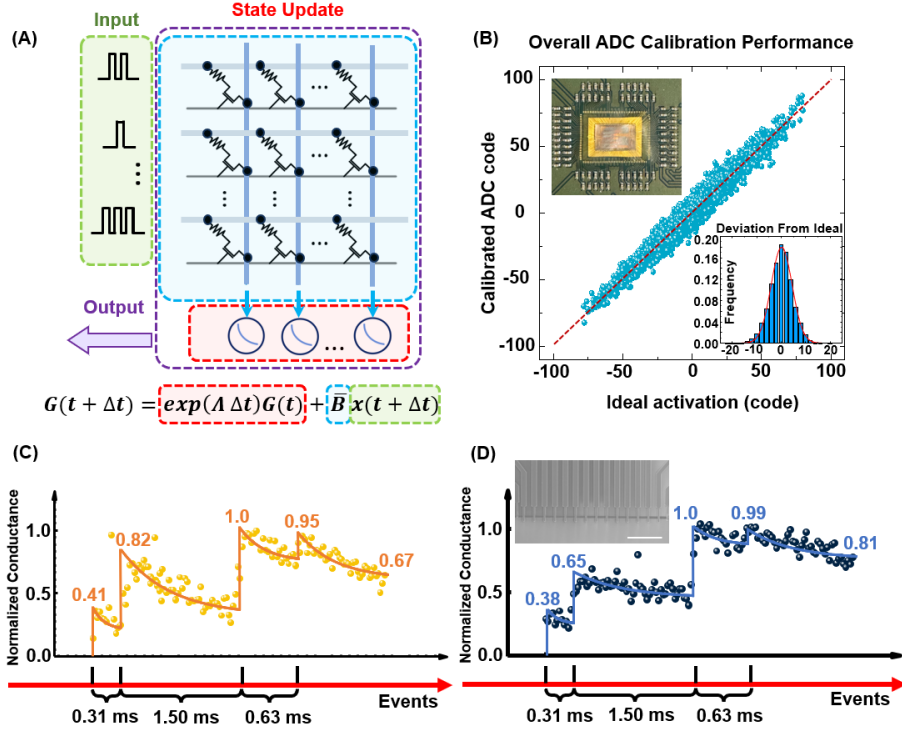


Figure 2. Hardware implementation of an SSM block. (A) Schematic of the proposed hardware implementation. The state update function, shown at the bottom, is physically realized through an RRAM crossbar array that performs VMM operation between the $\bar{B}$ matrix (blue) and the input vector (green), and an array of SNs that stores the state and perform the diagonalized state evolution functions (red). These functions directly map the correspondingly colored operations in the state update function. (B) VMM operations between the $\bar{B}$ matrix and the input vector implemented in an RRAM CIM chip. The experimentally measured VMM outputs from the RRAM CIM chip (dots) are plotted against ideal values (line). Upper inset: photo of the CIM chip. Lower inset: deviation of the measured data from the ideal VMM values due to device variations. A standard deviation of 4.6 LSB was measured. (C-D) State evolutions implemented in WO_x memristors with short-term memory, showing experimentally measured memristor conductance evolutions (dots) vs. expectations obtained from Eq. 8 (lines), for two required decay rates of 0.35 ms^{-1} (C) and 0.2 ms^{-1} (D). Each memristor act as an SN whose conductance represents a component of the state and performs state evolutions. The inset in (D) shows a scanning electron microscope (SEM) image of the fabricated WO_x memristor array. (scale bar: $20 \mu\text{m}$).⁹

Following your suggestions in this comment and in comments 8(v), we have re-organized the text, added detailed discussions on the embedding layer, moved the Hardware Implementation section to be immediately after the Architecture overview section, and expanded the Hardware Implementation section to include more details. We have also removed the term “reservoir node” to avoid confusion, following your comments in Comment 4. The revised text is copied below:

In p4

‘The channel-wise embedding layer, shown in Fig. 1(B), maps an event into a D -dimensional vector based on the event’s channel index. Since the embedding matrix $W_{\text{embedding}} \in \mathbb{R}^{(J \times D)}$, where J is the number of input channels, is static after training, it can be physically stored in an RRAM crossbar array and embedding is performed by reading out the values in the row corresponding to j_m when spike m arrives. Embedding is a common pre-processing technique used in modern models, e.g. language models. In this case, by mapping each channel to a distinct embedding vector, we endow every channel with a unique feature representation, enabling the SSM blocks to process event streams within a unified feature space.’

In p5

‘We note that Eq. (2) can then be efficiently implemented in analog circuits where the 2^{nd} term corresponds to a VMM function and can be implemented using an RRAM crossbar for the current input at $t + \Delta t$, while the 1^{st} term can be implemented in parallel using a group of devices (termed state nodes, SNs here) exhibiting decay dynamics, taking advantage of the diagonalized matrix Λ .’

In p6-p7 Hardware Implementation of SSM Block

‘Having detailed the mathematical framework of the proposed SSM, we now describe its physical implementation in a CIM-based hardware. We co-designed the hardware architecture to natively execute the model’s core operations, following the principles of performing computing directly through device and circuit physics. Specifically, we implement the state update equation by leveraging the native exponential decay of the internal state variable in memristor devices exhibiting STM effects³⁹. For example, prior studies have shown that when stimulated by an input, the WO_x -based memristor device conductance increases according to the input then spontaneously relaxes due to oxygen ion diffusion, resulting in STM behavior^{41,42}. Given that state parameters evolve independently (due to the use of diagonal state dynamics matrix) in an SSM block, the state can be represented by an array of such vector decay dynamics can be parallelized and directly implemented using STM memristors (SNs).

Specifically, the state update equation inside an SSM block (Eq. (2)) can be implemented as:

$$G(t + \Delta t) = \exp(\Lambda \Delta t)G(t) + \bar{B}x(t + \Delta t). \quad (8)$$

Here $G(t) = [g_0(t), g_1(t) \dots g_h(t)]$ is the collection of the STM memristor's conductance at time t and represents the hidden state vector, $\Lambda = \lambda I$ and λ is the layer-wise fixed decay constant, Δt is the time interval since the previous event, $\bar{B}$ is the input projection matrix and $x(t + \Delta t)$ is the input at time $t + \Delta t$ when the current spike arrives. We note that this equation can be physically implemented using an array of STM memristors that act as SNs to store the state and implement the state decay dynamics (1st term in Eq. (8)), and an RRAM crossbar that implements the VMM operation between the $\bar{B}$ matrix and the input (2nd term in Eq. (8)), as shown in Fig. 2(A). The proposed hardware thus directly implements the state update equation (Eq. 8), which in turn corresponds to the state equation Eq. 2.

We tested this implementation experimentally using an RRAM CIM chip that performs the VMM operations and a group of WO_x memristors that exhibit STM behaviors, respectively. The inset of Fig. 2B shows a photograph of the RRAM CIM chip for the VMM operations. This chip was fabricated using a 65 nm CMOS process with four integrated 64×64 RRAM arrays, which are logically combined to form an equivalent 128×128 array. Each RRAM device in the array is based on a one-transistor-one-resistor (1T1R) structure with a physical size of 200 nm^2 ⁴³. Each array is connected with integrated 8-bit digital-to-analog converters (DACs) for input encoding and 8-bit analog-to-digital converters (ADCs) for output readout. The measured VMM outputs from the RRAM CIM chip are shown in Fig. 2(B) (dots), along with the ideal expected values (line).

The measured output distribution (bottom inset, Fig. 2(B)) due to device variations is characterized by a standard deviation σ of 4.6 LSB (Least Significant Bit), corresponding to a Normalized Root-Mean-Square Error (NRMSE) of 1.80%. The high consistency between measured and ideal values and low NRMSE verify that the RRAM CIM configurations can accurately performing the VMM operation required for the $\bar{B}x(t + \Delta t)$ term in Eq. 8. More details of the RRAM CIM chip can be found in Supplementary Information.

The VMM output is then applied to the SNs to implement the exponential state evolution (1st term in Eq. (8)). Fig. 2(C-D) shows the state evolution function implemented with WO_x memristors that exhibit STM behaviors, where the $\exp(\Lambda \Delta t)G(t)$ function is passively implemented based on the native conductance decay in between inputs. The application of the VMM output updates the decayed conductance values, completing the state transition by naturally summing the two terms of Eq. 8 in the STM memristor devices.

To match the required decay profiles, WO_x devices with different oxide thicknesses were fabricated by tuning the rapid thermal processing (RTP) annealing duration (Methods),

following prior studies^{41,44}. Longer annealing times yield thicker oxide layers and higher oxygen vacancy densities, which in turn slow the relaxation process due to more persistent conductive filament paths. The tunable decay behavior quantified in Supplementary Information Figure S3.

Fig. 2(C) and 2(D) show experimentally implemented Eq. 8 with two target time constants. Strong agreement between measured outputs (scatter points) and the expected SSM-based evolution (solid lines obtained by simulation of Eq. (2)), confirming the viability of this approach for hardware SSM inference.

By fixing the decay constant to a single value in a block, only six distinct decay profiles—one per layer—are required for hardware implementation of the proposed SSM model. This significantly reduces the physical implementation challenges as precise control of the time constant τ to a different target value for every individual memristor device is impractical during fabrication.

The updated state from each SN is then read out and used to perform the positional non-linear gating and residual connection operations as described in Eq. 6 and Eq. 7. These operations, including the GELU and sigmoid activation functions, are implemented digitally using Look-Up Tables (LUTs) that store pre-computed values. The output is then passed to the next SSM block as a discrete input. This entire process, from event reception to SN state update, demonstrates the tight coupling between the mathematical formulation and physical device dynamics, enabling a highly efficient and clockless implementation of the SSM.'

We have added Figure S1 in the Supplementary Information (SI) that details calibration procedure used to correct for ADC and channel-level non-idealities in the RRAM CIM chip's output, explaining how we handle hardware imperfections:

'S1. Calibrating the output of RRAM CIM chip

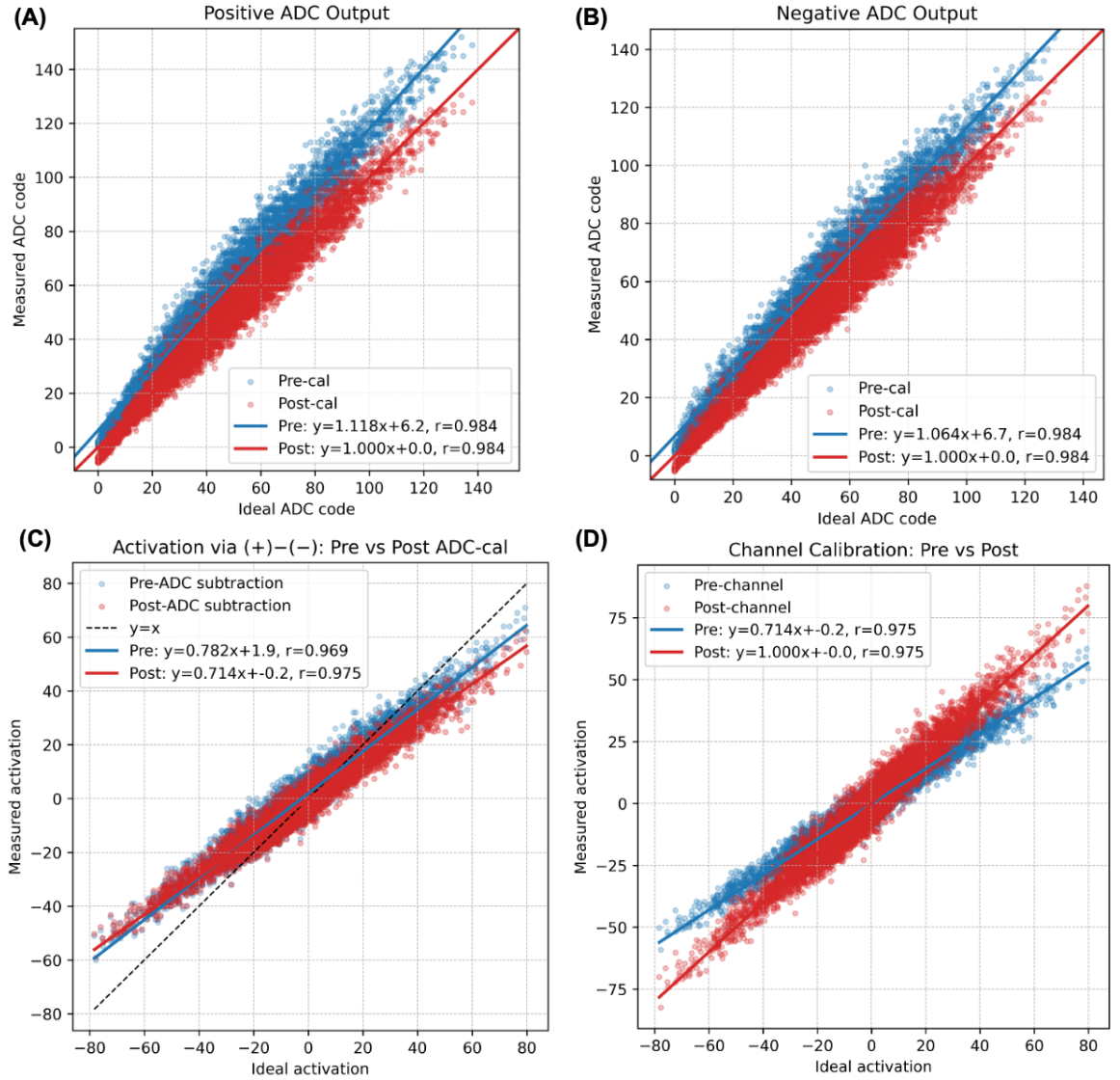


Fig. S1. Minimizing systematic errors through a two-stage calibration process. (A, B) Pre-calibration (blue) and post-calibration (red) outputs for the positive and negative columns through ADC, respectively, demonstrating the correction at the individual ADC-level (Stage I). (C) The combined activation after Stage I (ADC-level calibration and subtraction). While the correlation is much improved, some residual deviation from the ideal identity line (dashed black) remains. (D) The final output after Stage II (channel level calibration), which aligns almost perfectly with the ideal $y=x$ relationship.

The analog nature of compute-in-memory (CIM) systems introduces several sources of error. Non-idealities in the RRAM devices and readout circuitry lead to a discrepancy between the ideal and measured VMM outputs. To address these cumulative errors, we employ a two-stage linear calibration procedure: an ADC-level calibration followed by a channel-level activation calibration.

ADC-level Calibration: The first stage corrects errors at the individual ADC. Since our system uses a differential mapping scheme, where positive (W^+) and negative (W^-)

components of a weight are mapped to separate physical columns, the corresponding positive and negative ADC channels are calibrated independently. For each column c (either positive or negative), we perform a linear regression to model the relationship between the measured raw ADC codes (y_c) and the ideal codes (x_c), such that $y_c \approx s_c x_c + b_c$. The transformation is then applied to correct the raw output, yielding the ADC-calibrated code $\widehat{y}_c$: $\widehat{y}_c = a_c y_c + d_c$ where $a_c = \frac{1}{s_c}$ and $d_c = -\frac{b_c}{s_c}$. This step aligns the output of each physical channel to its ideal value as shown in Fig S1 (A) and (B). The ADC-calibrated activation is then obtained by subtracting the calibrated negative column's output from the calibrated positive column's output (Fig S1 (C)).

Channel-level Calibration: The second stage corrects remaining channel-level (the difference of the calibrated positive column and the corresponding negative column) errors. We regress the ADC-calibrated channel activation ($\widehat{y}_k$) for a given channel k on the final ideal activation (x_k), modeling the relationship as $\widehat{y}_k \approx \alpha_k x_k + \beta_k$. Following the same principle, we get calibrated output $\widetilde{y}_k = A_k \widehat{y}_k + B_k$ where $A_k = \frac{1}{\alpha_k}$ and

$$B_k = -\frac{\beta_k}{\alpha_k}.$$

These two sequential linear corrections can be mathematically fused into a single operation for direct physical hardware calibration. By substituting the Stage 1 equations into the Stage 2 equation, we derive a single transformation from the raw ADC codes ($y_{c,pos}$, $y_{c,neg}$) to the final output ($\widetilde{y}_k$):

$$\begin{aligned} \widetilde{y}_k &= A_k \widehat{y}_k + B_k = A_k (\widehat{y_{c,pos}} - \widehat{y_{c,neg}}) + B_k = A_k [(a_{c,pos} y_{c,pos} + d_{c,pos}) + \\ & (a_{c,neg} y_{c,neg} + d_{c,neg})] + B_k = (A_k a_{c,pos}) y_{c,pos} - (A_k a_{c,neg}) y_{c,neg} + \\ & [A_k(d_{c,pos} - d_{c,neg}) + B_k] \end{aligned}$$

This fused equation provides the exact target parameters for physically tuning the ADC hardware: The W^+ ADC's gain is set to $G_{pos} = A_k a_{c,pos}$, and the W^- ADC's gain is set to $G_{neg} = A_k a_{c,neg}$ and The total differential offset is set to $O_{diff} = A_k(d_{c,pos} - d_{c,neg}) + B_k$. In practice, this allows us to program these fused coefficients directly into the circuitry (e.g., by adjusting ADC reference voltages for gain and comparator biases for offset). This corrects all linear errors at the point of conversion, achieving the final calibrated result shown in Fig. S1 (D) and Fig. 2 (B) without the overhead of sequential digital post-processing.'

39. Chang, T., Jo, S.-H. & Lu, W. Short-Term Memory to Long-Term Memory Transition in a Nanoscale Memristor. *ACS Nano* **5**, 7669–7676 (2011).

41. Chang, T., Jo, S.-H. & Lu, W. Short-Term Memory to Long-Term Memory Transition in a Nanoscale Memristor. *ACS Nano* **5**, 7669–7676 (2011).
 42. Du, C. *et al.* Reservoir computing using dynamic memristors for temporal information processing. *Nat Commun* **8**, 2204 (2017).
 43. Wu, Y. *et al.* Bulk-Switching Memristor-Based Compute-In-Memory Module for Deep Neural Network Training. *Advanced Materials* **35**, 2305465 (2023).
 44. Chang, T. Tungsten Oxide Memristive Devices for Neuromorphic Applications. (2012).
2. It is not clear whether the results in Table 1 and 2 were obtained through simulation or hardware measurement. Please clarify in both caption and main text. It is hard to “imagine” the deployment of millions of parameters onto the proposed system without knowing the implementation details. If it was done by simulation, please provide full details about the simulation model, e.g. how was the noise modelled?

Response:

Thank you for pointing out this ambiguity. We have clarified in the Model Training and Performance Analysis section and in the captions that the performance results in Tables 1 and 2 are obtained at the software level.

Now we have clearly stated that:

In p8

‘We now detail the training procedure used to determine the model's optimal parameters. It is important to clarify that this training process, as well as the performance analysis in the following sections, is conducted at the software level.’

In p13

‘To better characterize our model's behavior prior to hardware mapping, we performed several analyses at the software level. Fig. 3 (A-D) shows how the classification accuracy evolves as we vary the number of stacked SSM blocks (model depth) for different event-driven datasets.’

And the captions for Table 1 and Table 2 are changed to:

In p11

‘Table 1: Comparison of our SSM model to the state-of-the-art SNN based models on Spike Audio Datasets in software-level’ and

In p12

‘Table 2: Comparison of our SSM model to other leading models on the DVS128 Gesture and DVS128 Lips datasets in software-level.’

This software level analysis serves to establish the effectiveness and parameter efficiency of our proposed hardware-friendly SSM algorithm before mapping it to the physical hardware.

After these software-level analyses that prove the function and accuracy of the proposed SSM models, we also performed analysis on hardware implementations including the effects of noise and device non-idealities. These discussions were included in the original text in page 14 but have been revised to enhance clarity. The revised discussions are copied below

In p14

‘We next evaluate the model implementation on CIM hardware using the SSC and DVS-Gesture datasets. To align the SSM model with analog CIM hardware constraints, all weights and activations are quantized to 8-bit integer (INT8) using quantization-aware training (QAT)^{64,65}. This step causes no accuracy degradation for the DVS-Gesture model (remaining at 97.3%) and only a minimal 0.3% drop for the SSC model (from 84.7% to 84.4%).

We then studied the effect of device non-idealities by introducing random hardware variations to the INT8 parameters (see Method) based on the measured device variation effect data in Fig. 2(B-D). This resulted in only a modest accuracy decline for both models without any re-training or other mitigation techniques: the SSC accuracy dropped 2.39% to 82.01%, and the DVS-Gesture accuracy dropped 2.2% to 95.1%. These results support the resilience of the proposed model for practical analog hardware implementations, likely due to the use of only fully-connect layers and the shallow model architecture. Future implementation of techniques such as variation-aware-training⁶⁵ is expected to further improve the hardware accuracy.’

To better explain how we model the noise in the hardware, we added a new section in the Method called Noise Modelling:

‘To evaluate the model's robustness against hardware non-idealities, we conducted a noise- injection simulation following the Quantization-Aware Training (QAT) process. In

this simulation, we injected noise after each computational step that is mapped to the physical hardware, namely VMM and the state update in the SNs. This injected noise was modeled using a Gaussian distribution ($\mathcal{N}(\mu, \sigma^2)$). The parameters for this distribution (mean μ and standard deviation σ) were not arbitrarily chosen. Instead, they were empirically derived directly from the experimental characterization data of our fabricated CIM chip and SN devices. This methodology ensures that our simulation accurately reflects the analog noise profile inherent in the physical system.'

64. Jacob, B. et al. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. in 2704–2713 (2018).

65. Wang, Q., Park, Y. & Lu, W. D. Device Variation Effects on Neural Network Inference Accuracy in Analog In-Memory Computing Systems. *Advanced Intelligent Systems* **4**, 2100199 (2022).

3. Following the last comment, please clearly present the data source throughout the manuscript. It is known the figure 3(C), (D) and S2 were experimentally measured, while fig 4(B) was simulated by TSMC PDK. What about other data, e.g. Fig 2? For the above reasons, it is hard to tell whether the results are technically sound.

Response:

Thank you for the suggestion. We have revised the manuscript to clearly state the source of all data.

The performance analysis in Figure 3 (previously Figure 2) is based on software level simulations as clarified in the section "Model Training and Performance Analysis". We have stated this in the revised text:

In p13

'We performed several additional analyses at the software level to analyze the model behavior.'

Additionally, we have shown the data measurement for Fig. 2(B) in the SI (refer to the response to the first comment). We have made revisions throughout the text to make it clear which data was obtained in software and which obtained in experimentally fabricated devices.

4. The authors are mixing multiple concepts, which may hinder reader comprehension. The term Reservoir Node should originate from reservoir computing. The authors

previously published reservoir computing works with the same device WO_x and the same decay property (ref. 58). As the decay factor Λ in eq. 2 and 9 are determined by the WO_x devices, it is fixed and less likely to be trained. Therefore the definition appears to be in line with reservoir computing. In fact, reservoir computing can be considered a SSM as the reservoir layer produces a state space. The authors should clarify the necessity of this new terminology and discuss how their SSM-based approach different from reservoir computing frameworks.

Response:

Thank you for your valuable comment. While we leverage device physics similar to that used in reservoir computing, our method is fundamentally different from reservoir computing. In particular, the SSM model is trained end-to-end including the state dynamic governed by the decay parameter Λ , whereas in RC systems only the output (or "readout") layer needs to be trained and the internal reservoir is not trainable. Although the decay constants are fixed in the proposed SSM during inference to align with hardware physics, their values are not random. Instead, they are systematically determined by first training them as learnable parameters and then fixing them to a layer-wise average that proved effective for the task. This principled optimization of the model's temporal memory properties stands in contrast to the fixed, randomly generated dynamics of a conventional reservoir.

We do agree with the reviewer that the use of term 'reservoir node' (RN) is confusing since it implies a random, fixed configuration where in the proposed SSM the target decay constant is obtained through end-to-end training instead. To avoid such confusion, we have removed the term 'reservoir node' (RN) throughout the text. Also, we have added a new paragraph in the Discussion section to explicitly address this point:

In p16

'We note the intrinsic short-term memory of physical devices like WO_x has also been used in networks such as reservoir computing (RC)⁴² to process temporal information. However, the proposed SSM model is fundamentally different from reservoir computing. In particular, the SSM model, including the state dynamic governed by the decay parameter Λ , is trained end-to-end, whereas in RC systems only the output (or "readout") layer needs to be trained and the internal reservoir is not trainable. Although the decay constants are fixed in the proposed SSM during inference to align with hardware physics, their values are not random. Instead, they are systematically determined by first training them as learnable parameters and then fixing them to a layer-wise average that proved effective for the task.'

42. Du, C. *et al.* Reservoir computing using dynamic memristors for temporal information processing. *Nat Commun* **8**, 2204 (2017).

5. The training process could be described in more detail. Please provide more details about the implementation of BPTT and how it handles the noisy dynamical behavior of WOX and RRAM? Are they differentiable? Also, it would be helpful to show the loss curve for the tasks.

Response:

Thank you for the valuable suggestions. We have expanded the description of our training process in the Methods section under a new sub-header "Parallelized BPTT". We clarify that we use a parallel associative scan algorithm, which reduces the complexity from $O(L)$ to $O(\log L)$ for long sequences while maintaining full gradient flow:

‘Parallelized BPTT

We train the model using backpropagation through time, computing gradients with respect to all temporal dependencies via automatic differentiation. The forward pass implements the discrete-time SSM recurrence $h(t + \Delta t) = \bar{A}(\Delta t)h(t) + \bar{B}x(t + \Delta t)$ for sequences of length L . Rather than computing this recurrence sequentially, we leverage a parallel associative scan algorithm that exploits the associativity of linear recurrences through a operator $(A_i, b_i) \circ (A_j, b_j) = (A_j \cdot A_i, A_j \cdot b_i + b_j)$. This formulation enables simultaneous computation of all hidden states via a balanced binary tree reduction, achieving $O(\log L)$ parallel time complexity compared to $O(L)$ for sequential evaluation given sufficient parallel processors⁷⁴. During the backward pass, automatic differentiation through the scan operation computes gradients for all SSM parameters—including state matrix A , input projection B , output projection C , and learnable discretization timesteps—while preserving the same logarithmic complexity. This parallelized BPTT implementation allows efficient gradient-based optimization on long event sequences while maintaining full temporal credit assignment across all time steps.’

74. Blelloch, G. E. Prefix sums and their applications. in 1294199 Bytes (Carnegie Mellon University, 2004). doi:10.1184/R1/6608579.V1.

Loss curves for the training are now added in the SI part:

‘S5. Loss Curve for Different Tasks

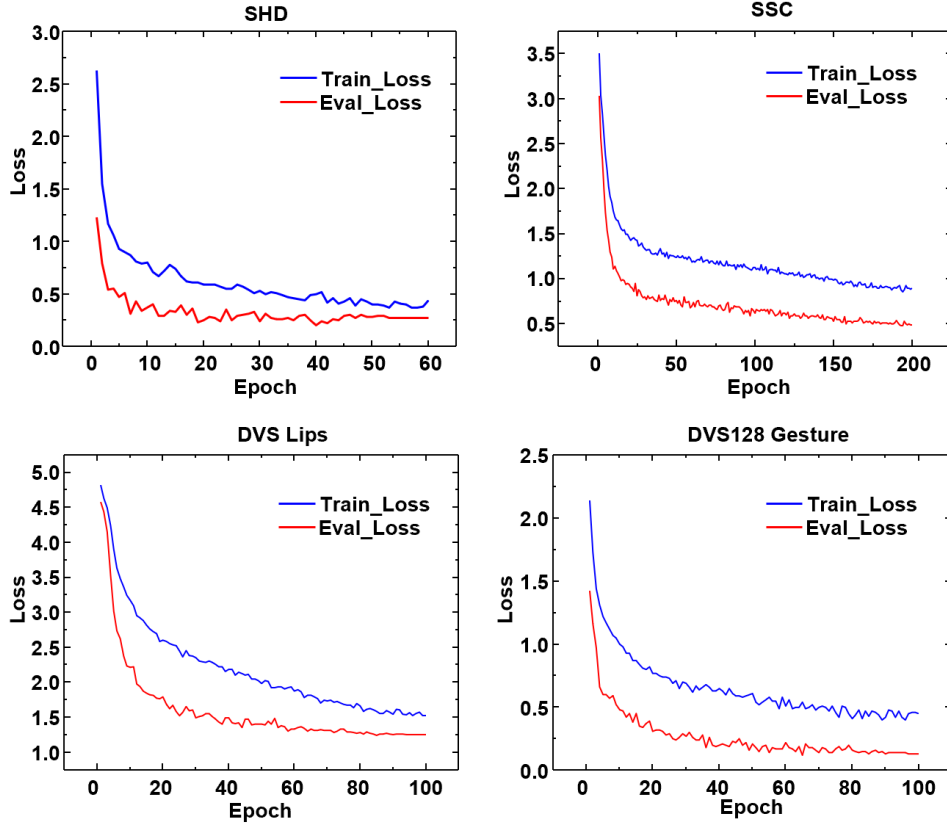


Fig S4. Training loss (blue line) and evaluation loss (red line) versus training epochs for the software-level model for the four event-driven datasets (Spiking audio tasks SHD and SSC, and the event-based vision tasks DVS Lips and DVS128 Gesture).’

Regarding noise, our current training process does not directly model the hardware non-idealities during training phase. The training is performed at the software level to find optimal parameters. Even without explicitly incorporating device non-idealities during training, the model’s robustness to hardware noise was proven post-training, as discussed in the response for your comment 2. We agree that incorporating device non-idealities into the training process, such as using variation-aware training methods can further improve model performance in the presence of device non-idealities, and we have added this in the revised text:

In p19

‘We next evaluate the model implementation on CIM hardware using the SSC and DVS-Gesture datasets. To align the SSM model with analog CIM hardware constraints, all weights and activations are quantized to 8-bit integer (INT8) using quantization-aware training (QAT)^{64,65}. This step causes no accuracy degradation for the DVS-Gesture model

(remaining at 97.3%) and only a minimal 0.3% drop for the SSC model (from 84.7% to 84.4%).

We then studied the effect of device non-idealities in analog hardware by introducing random hardware variations to the INT8 parameters (see Method) based on the measured device variation effect data in Fig. 2(B-D). This resulted in only a modest accuracy decline for both models without any re-training or other mitigation techniques: the SSC accuracy dropped 2.39% to 82.01%, and the DVS-Gesture accuracy dropped 2.2% to 95.1%. These results support the resilience of the proposed model for practical analog hardware implementations, likely due to the use of only fully-connect layers and the shallow model architecture. Future implementation of techniques such as variation-aware-training⁶⁵ is expected to further improve the hardware accuracy.'

64. Jacob, B. et al. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. in 2704–2713 (2018).

65. Wang, Q., Park, Y. & Lu, W. D. Device Variation Effects on Neural Network Inference Accuracy in Analog In-Memory Computing Systems. *Advanced Intelligent Systems* **4**, 2100199 (2022).

6. The authors claimed fully asynchronous processing. But could the authors clarify how the claimed asynchronicity is maintained given that the ADC/DAC likely require clocks? The extent of asynchronicity needs to be clarified.

Response:

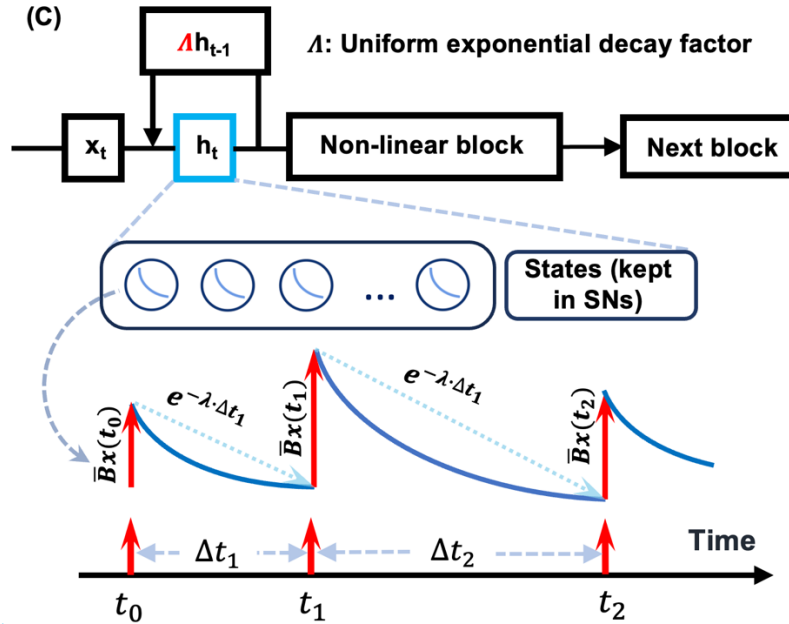
Thanks for your valuable suggestion. We have revised the text in the Introduction section to explain how asynchronous processing is achieved in the proposed system:

In p5

‘As illustrated in Fig. 1(C), When no inputs are applied (e.g., during intervals Δt_1 and Δt_2), the state evolve passively and natively through the short-term memory property of the SNs, implementing the exponential decay (e.g., $e^{-\Delta t_1}, e^{-\Delta t_2}$) without the need of clocks or synchronization. Active computation is needed only when an input arrives (e.g., at time t_1, t_2), where the projected input term (e.g., $\bar{B}x(t_0), \bar{B}x(t_1), \bar{B}x(t_2)$) is computed via VMM and added to the current state, making the system operation fully event-driven. The updated state is then read out, further processed through a non-linear block, and applied as an input event to trigger the next block's computation. By leveraging physics, including Ohm's law, Kirchhoff's current law, and internal device dynamics, this

approach achieves highly parallel and energy-efficient computation and is compatible with asynchronous inputs.'

Fig. 1(C) is also attached below:



(C) Operational schematic of a single SSM block (upper). The state of the SSM is represented by those of individual state nodes (SNs) (middle) in the proposed implementation. The state in an individual SN decays exponentially following a trained decay time constant. When an input x arrives, it is projected onto the state space through $\bar{B}x$ and is added to the current state. The updated state h is then read out, passed through some non-linear functions and sent to the next block. Note actions (i.e. computations) are only performed when an event arrives, allowing asynchronous operations.'

We agree that the use of ADCs and DACs still require clocks. However, the system's operation is event-driven, and computations are only performed when an event arrives, as discussed above. Since the events are significantly less frequent (e.g. kHz scale) vs the clock frequency (40MHz), the system operation is considered asynchronous. We agree that future developments that remove ADCs and DACs, i.e. making the system fully analog, will make the system even more efficient. We added the following discussions in the Discussion section.

In p15-16

'However, we note that the estimate was based on ADCs and other digital components constantly operating at 40MHz due to the limitations of the current CIM chip. In practice,

when the operations are truly operated in an event-driven fashion, the ADCs will not be active all the time and actual power consumption can be significantly lower. The power can be further improved by eliminating the analog-digital transitions to implement all operations in the analog domain, which is compatible with SSM based approaches.'

7. In Line 374, the authors didn't explain how to measure the power of RN devices and CIM chip. This is an important part that would benefit from more details.

We thank the reviewer for this important comment. We have added a new subsection called 'Power Analysis' in the Method section to include a detailed breakdown of our power measurement methodology for all the components:

'Power Analysis

The power for the CIM chip, which performs the VMM, was measured directly on the PCB board. We measured the power consumption while the chip was performing a typical VMM operation at its operational clock frequency of 40 MHz. Energy consumption is calculated as Power Consumption per array $\times$ number of arrays needed and the measured power consumption per array is about 0.028mW. It is important to note that, due to the limitations of the current test chip, this measurement was conducted under a global clock, and the clock frequency is much higher than the requirement of the dataset (After pooling, SSC dataset will have roughly 5000 events/second, corresponding to operation at the kilohertz level). We therefore believe that a future implementation using a fully event-driven CIM chip would exhibit even lower power consumption, as it would theoretically only perform VMM computations when an input event arrives, thus fully leveraging data sparsity.

Power consumption of the STM memristor array is dynamic and event-driven. Therefore, we estimated the total energy E_{SN} by summing the energy of each individual inputs. We used the methodology from Yoo et al.⁷⁶, as shown in the following equation:

$$E_{SN} = \sum_{n=1}^N V_{pulse}^2 * G(x_n, \gamma_n, p_n) * t_{pulse} \quad (10)$$

where $G(x_n, \gamma_n, p_n)$ represents the conductance of the specific SN device being addressed. For each SN updates, which corresponds to an input spike $x(t+\Delta t)$ in Eq. 8, a burst of N 1.4 V / 50 μ s pulses were applied on the memristor, where the number of pulses N is proportional to the VMM output.

Data from the remaining components are obtained from synthesized circuits using TSMC 28nm node, following an approach by Wang et al.²⁴ Inside each block, positive and negative

weight values are mapped onto separate RRAM arrays. The 8-bit input activations are applied in a bit-serial fashion using two voltage pulses, each representing a 4-bit value (16 discrete levels). The resulting analog currents from RRAM arrays are digitized by 8-bit ADCs and subsequently shift-added digitally to obtain an 8-bit partial sum, representing the output activation values that are applied to the corresponding SN devices. In our model, the layer at the front uses a smaller state dimension and the layers at the back use a larger state dimension to allow for richer information compressed across time. Each SSM block also require two LUTs to implement the non-linear functions, namely, GELU and Sigmoid. The power of these nonlinear functions are similarly simulated. Given that these functions operate on the 8-bit quantized values from ADCs, each non-linear operation can be efficiently realized using a compact LUT containing only 2^8 (256) pre-computed entries. The power for one LUT is estimated to be 0.25 mW at 40 MHz.'

In p15-p16

'The total power consumption for our system is estimated to be 34 mW and the energy distribution across system components is shown in Fig. 4(B) (see Method). The 8-bit ADCs dominate the power budget, accounting for 43.7% of total power usage. Analog power consumption in the RRAM arrays follow, consuming 30.3%. SN devices for update behaviors consume 0.4%. However, we note that the estimate was based on ADCs and other digital components constantly operating at 40MHz due to the limitations of the current CIM chip. In practice, when the operations are truly operated in an event-driven fashion, the ADCs will not be active all the time and actual power consumption can be significantly lower. The power can be further improved by eliminating the analog-digital transitions to implement all operations in the analog domain, which is compatible with SSM based approaches.'

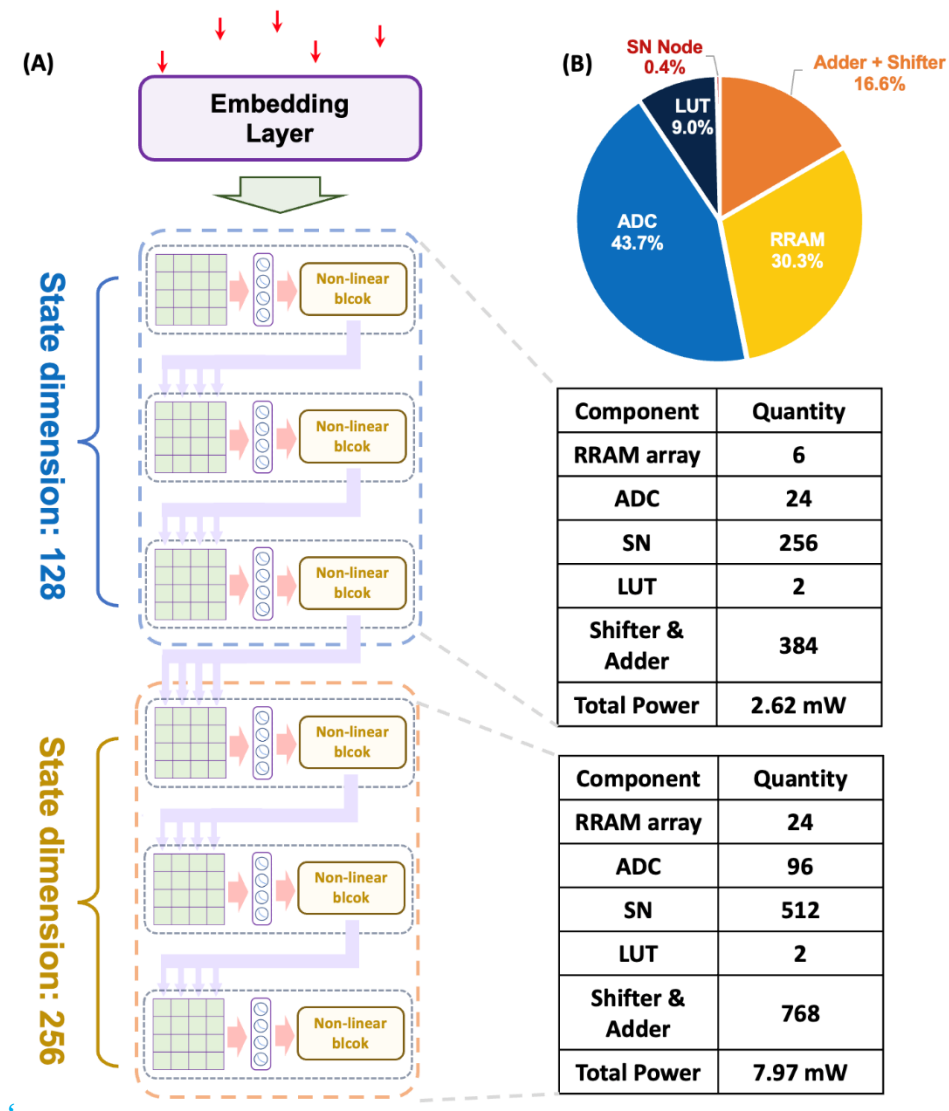


Figure 4. (A) Architecture of the hardware implementation for the SSC dataset, including a list of hardware components for different stages. Arrows represent data flow within an SSM block and between SSM blocks. (B) Power distribution across key hardware components for the SSC dataset implementation.'

24. Wang, X. et al. TAICHI: A Tiled Architecture for In-Memory Computing and Heterogeneous Integration. *IEEE Transactions on Circuits and Systems II: Express Briefs* **69**, 559–563 (2022).

76. Yoo, S., Lee, E. Y.-J., Wang, Z., Wang, X. & Lu, W. D. RN-Net: Reservoir Nodes-Enabled Neuromorphic Vision Sensing Network. *Advanced Intelligent Systems* **6**, 2400265 (2024).

8. Both figures and text are poorly organized. The main problems are:

(i). fig 1 (B) and (C) are somewhat unclear. Why the embedding vector is pointed to a straight line? It is not understood what exactly is the embedding. Both captions mention 'internal structure'. The current figures may illustrate some functionalities but the structure is not clearly depicted.

Response:

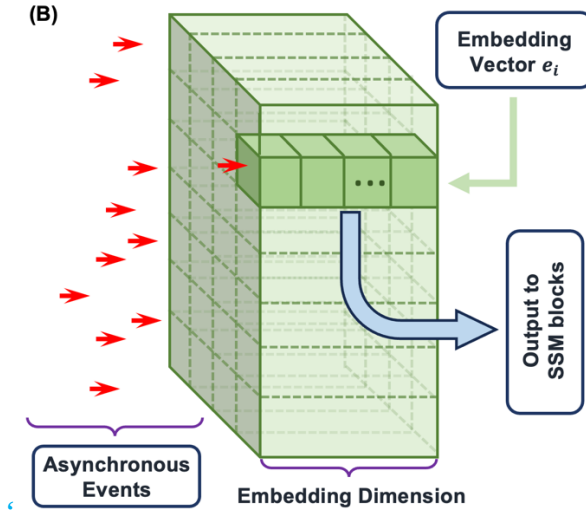
We thank the reviewer for this comment. In response to this comment, we have revised Figure 1 (also following suggestions in your first comment) and the accompanying descriptions to better illustrate both the functionality and the structure of our proposed system.

Specifically, we have made the following changes in the text to address the points raised:

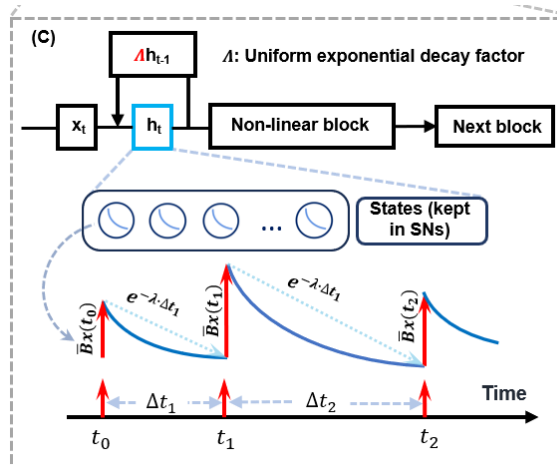
In p4

'Fig. 1 (A) shows the high-level model architecture, which is designed for asynchronous event processing. The input event-stream is an ordered set of events $E = \{(t_m, j_m)\}$, where t_m is the timestamp and j_m is the channel index of event m . In audio tasks, channel indices correspond to distinct frequency bands whereas in vision tasks, channels map one-to-one with pixel locations. The channel-wise embedding layer, shown in Fig. 1(B), maps an event into a D -dimensional vector based on the event's channel index. Since the embedding matrix $W_{embedding} \in \mathbb{R}^{J \times D}$, where J is the number of input channels, is static after training, it can be physically stored in an RRAM crossbar array and embedding is performed by reading out the values in the row corresponding to j_m when spike m arrives. Embedding is a common pre-processing technique used in modern models, e.g. language models. In this case, by mapping each channel to a distinct embedding vector, we endow every channel with a unique feature representation, enabling the SSM blocks to process event streams within a unified feature space.'

Also, we have the captions of the figures to better reflect the content. Now the captions are:



(B) (B) Functional diagram of the channel-wise embedding layer. An incoming spike activates the embedding layer which produces a D -dimensional vector depending on the event's channel index.



(C) Operational schematic of a single SSM block (upper). The state of the SSM is represented by those of individual state nodes (SNs) (middle) in the proposed implementation. The state in an individual SN decays exponentially following a trained decay time constant. When an input x arrives, it is projected onto the state space through $B\bar{x}$ and is added to the current state. The updated state h is then read out, passed through some non-linear functions and sent to the next block. Note actions (i.e. computations) are only performed when an event arrives, allowing asynchronous operations.'

(ii) Insufficient labeling in figures obscures essential experimental/design details. The authors should label all the terms. For instance, what are Δt_1 and Δt_2 in Fig 1(c)? What does every arrow means in fig 4(A)? In fig. 4(B), the power distribution percentages quoted in the text (lines 387-389) do not match those shown in Figure

4(B). And What is LUT? It has not been mentioned throughout the text. In table 2, the acc. for DVS128 is 63.5%, that is, different from the 62.8% in line 280. Please double check the paper.

Response:

Regarding Figure Labeling and Clarity:

For Δt_1 and Δt_2 in Fig. 1(c): Thank you for highlighting this. As you suggested, these terms needed clarification. As discussed in response to your Comment 1, we have re-drew Fig. 1(c) and revised the accompanying texts to make the description more clear. Specifically, the discussions of Δt_1 and Δt_2 have been added to p5, copied below:

In p5

‘As illustrated in Fig. 1(C), When no inputs are applied (e.g., during intervals Δt_1 and Δt_2), the state evolve passively and natively through the short-term memory property of the SNs, implementing the exponential decay (e.g., $e^{-\Delta t_1}, e^{-\Delta t_2}$) without the need of clocks or synchronization. Active computation is needed only when an input arrives (e.g., at time t_1, t_2), where the projected input term (e.g., $\bar{B}x(t_0), \bar{B}x(t_1), \bar{B}x(t_2)$) is computed via VMM and added to the current state, making the system operation fully event-driven. The updated state is then read out, further processed through a non-linear block, and applied as an input event to trigger the next block's computation. By leveraging physics, including Ohm's law, Kirchhoff's current law, and internal device dynamics, this approach achieves highly parallel and energy-efficient computation and is compatible with asynchronous inputs.’

For the arrows in Fig. 4(A): They represent data flow between the different functional units. We have revised Fig. 4(A) to better illustrate the system architecture, and accordingly modified the caption for Figure 4:

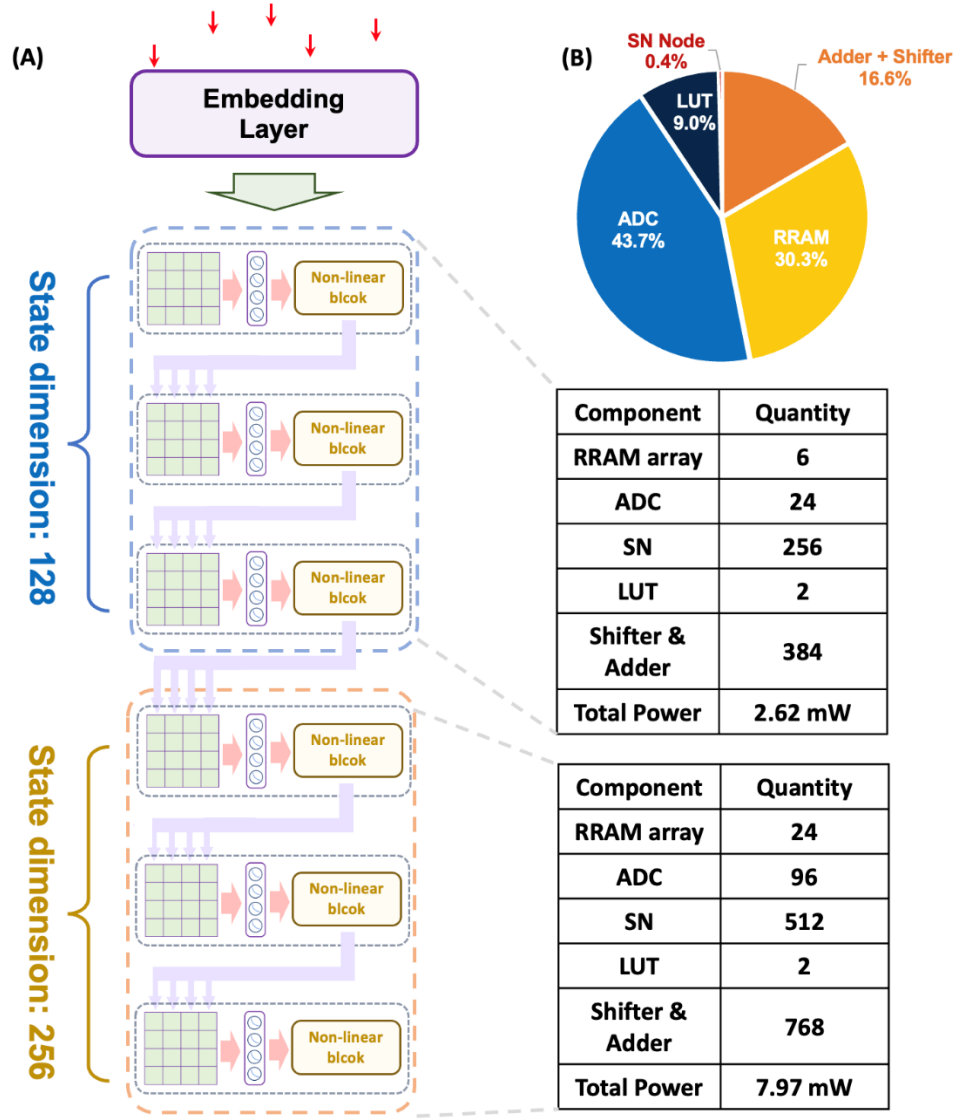


Figure 4. (A) Architecture of the hardware implementation for the SSC dataset, including a list of hardware components for different stages. Arrows represent data flow within an SSM block and between SSM blocks. (B) Power distribution across key hardware components for the SSC dataset implementation.’

For the term "LUT": We apologize for this omission. "LUT" stands for Look-Up Table. We have now defined this acronym and explained its function within our Hardware Implementation section in the main text:

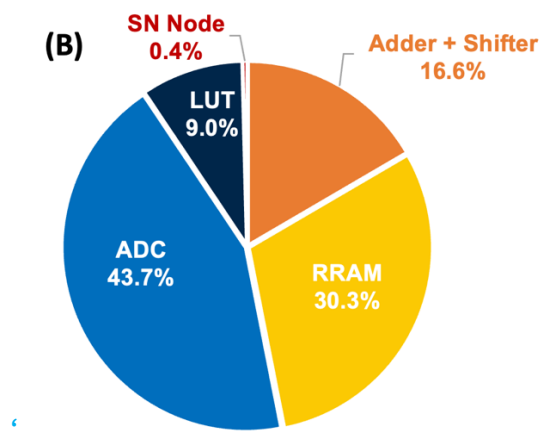
In p7-p8

‘The updated state from each SN is then read out and used to perform the positional non-linear gating and residual connection operations as described in Eq. 6 and Eq.

7. These operations, including the GELU and sigmoid activation functions, are implemented digitally using Look-Up Tables (LUTs) that store pre-computed values.'

Regarding Mismatched Values:

Power Distribution in Fig. 4(B): We sincerely thank you for catching this error. You are correct that the percentages in the text did not match the figure. This was an oversight from when we updated the text during manuscript writing but forgot to regenerate the corresponding figure. We have now corrected the pie chart in Fig.4(B) to align with the text, showing the correct power distribution: ADCs (43.7%), RRAM array (30.3%), and SNs (0.4%) as we have shown above. The Fig. 4(B) is attached:



(B) Power distribution across key hardware components for the SSC dataset implementation.'

Accuracy in Table 2 vs. Text: The accuracy for DVS128 Lips was indeed inconsistent. We have double-checked our results and corrected the value in the text which is 63.5%:

In p14

'Specifically, SHD improves from 94.7% to 95.7%, DVS128 Gesture from 96.6% to 97.3%, and DVS128 Lips from 61.1% to 63.5%.'

(iii) It is suggested to totally reorganize the Results and Methods section. The majority of line 121-145 looks like a figure caption. And there are too many method descriptions in the Result section. Please first clear up what information should be in the Results, Methods, figure caption and SI, and then rewrite accordingly.

Response:

Thanks for your comment. Following this comment and your comment 8 (v), we have reorganized the paper. In accordance with your advice, we have substantially reorganized the Results section to follow this logical structure. The paper now presents our work in the following order:

We begin by introducing our system architecture and the hardware-friendly SSM algorithm. Next, we detail the hardware implementation and present the experimental validation of its core components. Following this, we provide a comprehensive performance analysis of the model. Finally, we conclude the section with a direct comparison of our approach against state-of-the-art methods, placing the comparative tables (Table 1 and 2) at this concluding point to summarize our findings. We have also moved some detailed discussions to Methods to improve the flow.

We have also re-written the description of Figure 1 that substantially shortened the text and made it flow better. The revised text has been copied:

In p4

‘Fig. 1 (A) shows the high-level model architecture, which is designed for asynchronous event processing. The input event-stream is an ordered set of events $E = \{(t_m, j_m)\}$, where t_m is the timestamp and j_m is the channel index of event m . In audio tasks, channel indices correspond to distinct frequency bands whereas in vision tasks, channels map one-to-one with pixel locations. The channel-wise embedding layer, shown in Fig. 1(B), maps an event into a D -dimensional vector based on the event’s channel index. Since the embedding matrix $W_{embedding} \in \mathbb{R}^{J \times D}$, where J is the number of input channels, is static after training, it can be physically stored in an RRAM crossbar array and embedding is performed by reading out the values in the row corresponding to j_m when spike m arrives. Embedding is a common pre-processing technique used in modern models, e.g. language models. In this case, by mapping each channel to a distinct embedding vector, we endow every channel with a unique feature representation, enabling the SSM blocks to process event streams within a unified feature space.

The embedded event representations are then passed through a series of stacked SSM blocks...’

(iv) The error of Fig 3(A) should be quantified using e.g. NRMSE.

Response:

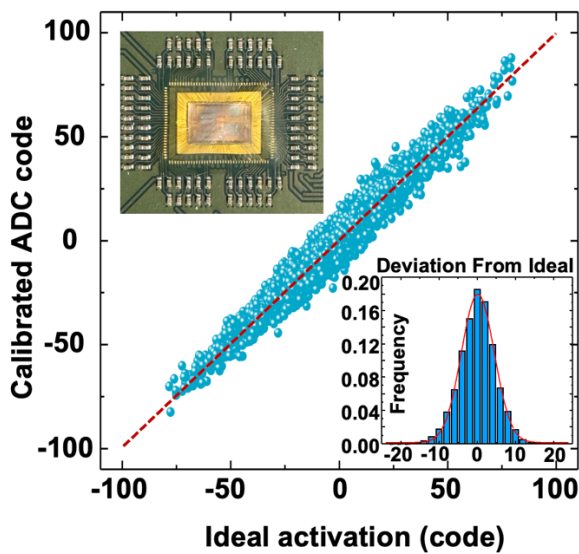
We thank the reviewer for the valuable suggestion. As requested, we have quantified the linearity error of the 8-bit ADC, and added the distribution data as an inset to current Fig. 2B (Previous Fig. 3A).

In p7

‘We tested this implementation experimentally using an RRAM CIM chip that performs the VMM operations and a group of WO_x memristors that exhibit STM behaviors, respectively. The inset of Fig. 2B shows a photograph of the RRAM CIM chip for the VMM operations. This chip was fabricated using a 65 nm CMOS process with four integrated 64×64 RRAM arrays, which are logically combined to form an equivalent 128×128 array. Each RRAM device in the array is based on a one-transistor-one-resistor (1T1R) structure with a physical size of 200 nm^2 . Each array is connected with integrated 8-bit digital-to-analog converters (DACs) for input encoding and 8-bit analog-to-digital converters (ADCs) for output readout. The measured VMM outputs from the RRAM CIM chip are shown in Fig. 2(B) (dots), along with the ideal expected values (line).

The measured output distribution (bottom inset, Fig. 2(B)) due to device variations is characterized by a standard deviation σ of 4.6 LSB (Least Significant Bit), corresponding to a Normalized Root-Mean-Square Error (NRMSE) of 1.80%. The high consistency between measured and ideal values and low NRMSE verify that the RRAM CIM configurations can accurately performing the VMM operation required for the $B\bar{x}(t+\Delta t)$ term in Eq. 8. More details of the RRAM CIM chip can be found in Supplementary Information.

(B) Overall ADC Calibration Performance



(B) VMM operations between the $\bar{B}$ matrix and the input vector implemented in an RRAM CIM chip. The experimentally measured VMM outputs from the RRAM CIM chip (dots) are plotted against ideal values (line). Upper inset: photo of the CIM chip. Lower inset: deviation of the measured data from the ideal VMM values due to device variations. A standard deviation of 4.6 LSB was measured.'

(v) The authors put the comparison table after fig 1. It is suggested to restructure the paper to follow a logical progression: problem -> algorithm -> hardware -> validation -> comparison.

Response:

Thank you for this valuable suggestion regarding the paper's structure. We would like to clarify that the current organization of the manuscript was drafted to follow the standard format for Nature Communications. This format places the main Results section immediately after the Introduction, followed by the Discussion and Methods.

Within this standard structure of the journal, we have restructured our paper following the logical progression you outlined:

Problem: This is established in the Introduction. We identify that common models (like SNNs with convolutional blocks) are architecturally mismatched for asynchronous data and CIM hardware. While SSMs are a powerful alternative, we define the core problem as their own critical hardware-mapping challenges: the high overhead of their complex-valued arithmetic and the difficulty of efficiently implementing their state dynamics on-chip.

Algorithm: This is presented first within the Results section, under the subsection "Model Architecture".

Hardware: This directly follows the algorithm, detailed in the "Hardware Implementation of SSM Block" subsection, which describes how the model is mapped onto the hardware.

Validation: This is covered in the subsequent "Model Training and Performance Analysis" and "Hardware System Performance Analysis" subsections. These sections validate the performance of our mapped model in terms of accuracy, power, and robustness.

Comparison: This is integrated within our performance validation. We provide comparisons of our method against other approaches (such as SNNs and synchronous methods) in Tables 1 and 2 and the accompanying text to benchmark our results. We also compare hardware results (including device variation effects) with software results in the "Hardware System Performance Analysis" subsection.

Minor:

Line 142: simply -> simplified

Response:

Thank you. We have changed it.

Line 326: double check the ref. 30. RN wasn't found here.

Response:

Thank you. Following your comment ..., we have removed the term "RN".

Line 374: system performance or power consumption?

Response:

Thank you. After arranging the text following your other comments, this section now discusses system hardware, performance (including variation effects), and power consumption, so we changed the title to "Hardware System Performance Analysis".

Line 40: Transformer block wasn't found in ref. 11,12 and 13.

Response:

We thank the reviewer for pointing out the inaccuracy in our citations. You are correct that ref. 11,12 and 13 in the original manuscript did not contain examples of SNNs incorporating Transformer blocks. This was an oversight on our part.

To correct this, we have updated our citations to better support our statement. We have now included a new, more relevant reference (Zhou et al., "Spikformer: When Spiking Neural Network Meets Transformer,") which explicitly introduces a Transformer-based architecture for Spiking Neural Networks:

In p2

'To improve model accuracy, SNNs often incorporate spatial processing blocks such as convolution or transformer blocks to help extract spatial features^{14–16}.'

14. Indiveri, G., Corradi, F. & Qiao, N. Neuromorphic architectures for spiking deep neural networks. in 2015 IEEE International Electron Devices Meeting (IEDM) 4.2.1-4.2.4 (2015). doi:10.1109/IEDM.2015.7409623.

15. Davies, M. et al. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. IEEE Micro 38, 82–99 (2018).

16. Zhou, Z. et al. Spikformer: When Spiking Neural Network Meets Transformer.
Preprint at <https://doi.org/10.48550/arXiv.2209.15425> (2022).

Reviewer #2:

This paper presents a methodology for implementing SSMs on CIM hardware, and holds significant potential by crossing over two emerging fields. In my view, this paper can enable significant impact. Some minor notes that could enhance the quality and technical accuracy of the manuscript are provided below. Note there are several references used to back up several points in this review. I do not expect the authors to include these citations.

Response:

Thanks for your valuable comments.

Introduction

- The authors state that the use of surrogate gradients is what compromises model accuracy. While this is broadly the accepted view and commonly cited reason, I'd contend that surrogate gradients are generally quite robust (especially in the era of QAT and extreme low-precision models that have been shown to scale up). I expect the main limiter is binarization of activations. This drives excessive information compression. This constraint is highlighted in Zhu et al.'s work in SpikeGPT (TMLR, 2024).

Response:

We thank the reviewer for this insightful comment. We agree with the reviewer's perspective that the original statement oversimplified the challenge of training SNNs. The surrogate gradients have proven to be increasingly robust, and that the binarization of activations could be a more fundamental limitation due to significant information compression.

To address this, we have revised this discussion in the Introduction section:

In p1-p2

'However, despite their potential, training deep SNNs can be challenging due to the non-differentiable nature of spike generation, which necessitates the use of surrogate gradient techniques for backpropagation⁹. While these techniques have proven increasingly robust^{10,11}, the inherent binarization of activations may still limit model performance, a constraint recently identified as a key factor for scaling up SNNs¹². Tradeoffs between different encoding methods and neuron models have also been found to affect model performance¹³.'

9. Eshraghian, J. K. *et al.* Training Spiking Neural Networks Using Lessons From Deep Learning. *Proceedings of the IEEE* **111**, 1016–1054 (2023).
10. Courbariaux, M., Bengio, Y. & David, J.-P. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. in *Advances in Neural Information Processing Systems* vol. 28 (Curran Associates, Inc., 2015).
11. Han, S., Mao, H. & Dally, W. J. Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding. Preprint at <https://doi.org/10.48550/arXiv.1510.00149> (2016).
12. Zhu, R.-J., Zhao, Q., Li, G. & Eshraghian, J. K. SpikeGPT: Generative Pre-trained Language Model with Spiking Neural Networks. Preprint at <https://doi.org/10.48550/arXiv.2302.13939> (2024).
13. Guo, W., Fouda, M. E., Eltawil, A. M. & Salama, K. N. Neural Coding in Spiking Neural Networks: A Comparative Study for Robust Neuromorphic Systems. *Front. Neurosci.* **15**, (2021).

- The paragraph that highlights the mismatch for convolutions and transformers for CIM makes a lot of sense. Buchel & Sebastian et al.'s work in "Efficient scaling of large language models with mixture of experts and 3D analog in-memory computing", Nat Comp Sci, 2025 is a good reference for this, where all Transformer/self-attention operations are delegated to digital circuitry. However, an additional challenge is that applying CIM to dense-layers alone does not address the primary bottleneck of Transformer ops for long sequence length. I.e., Transformers will dominate the computational cost relative to dense layers. This paragraph (or the start of the following paragraph) could be made more compelling by more clearly stating that SSMS overcome the memory-bottleneck of self-attention/KV-caching, making them a natural fit for CIM architectures.

Response:

Thank you for the valuable suggestions. We have revised the Introduction section to make this argument more compelling, as suggested. We now explicitly state that the core problem with Transformers is not only their incompatibility with stationary CIM arrays but also their memory-bottleneck of self-attention/KV-caching:

In p2

‘The attention mechanism, representing input-dependent weights, is also incompatible with stationary CIM arrays. Additionally, the use of Key-Value (KV) cache which increases quadratically with input sequence length and needs to be constantly updated also necessitates hybrid designs that delegate these operations to separate digital circuitry²⁷.

To overcome both the architectural mismatch with convolutions and the memory and scaling bottleneck inherent in Transformers, in this work...’

27. Büchel, J. et al. Efficient scaling of large language models with mixture of experts and 3D analog in-memory computing. *Nat Comput Sci* **5**, 13–26 (2025).

- There is some tightly related work where SSMS are deployed on neuromorphic/event-based hardware by Abreu et al. (Neuromorphic Principles for Efficient Large Language Models on Intel Loihi 2, ICLR SCOPE Workshop, 2025) & Zhu et al. (Scalable MatMul-Free Language Modeling on Neuromorphic Hardware, arXiv 2025). Neither uses CIM, and instead relies on digital, async accelerators.

Response:

We thank the reviewer for highlighting these highly relevant and concurrent studies. The papers by Abreu et al. and Zhu et al. are indeed important for contextualizing our research. We have added a sentence in the Introduction section to acknowledge these works and clarify the difference with our study:

In p3

‘...While recent works have successfully mapped SSMS onto digital neuromorphic processors^{37,38}, our approach represents a co-design methodology leveraging analog CIM to perform key operations, including state update, directly through device physics with potentially greater efficiency and scalability.’

37. Abreu, S., Shrestha, S. B., Zhu, R.-J. & Eshraghian, J. Neuromorphic Principles for Efficient Large Language Models on Intel Loihi 2. Preprint at <https://doi.org/10.48550/arXiv.2503.18002> (2025).

38. Zhu, R.-J. et al. Scalable MatMul-free Language Modeling. Preprint at <https://doi.org/10.48550/arXiv.2406.02528> (2025).

- I would encourage citing the original SSM+DL work by Voelker, Kajic & Eliasmith, "Legendre memory units: Continuous-time representation in recurrent neural networks", neurips, 2018, where referring to "grounded in control theory".

Response:

We thank the reviewer for this excellent suggestion. We have added this reference:

In p2

‘Grounded in control theory with theoretical guarantee to stably approximate continuous-time convolutional kernels, SSMs model the sequence through a latent state vector that evolves over time based on the current input and the previous state³⁰.’

30. Voelker, A., Kajić, I. & Eliasmith, C. Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks. in *Advances in Neural Information Processing Systems* vol. 32 (Curran Associates, Inc., 2019).

- The use of real-valued coefficients in the transition matrix is acceptable, and the use of non-dynamic/input-dependent transition matrices for the update is a fair fit for CIM, though the dynamic transition matrix is a big part of what boosts performance in recurrent architectures. I think it is fair to state frankly that this is a limitation, although, if it was delegated to digital computation, it's a tiny fraction of the overall compute.

Response:

We agree that the performance of modern SSMs, such as Mamba, is significantly boosted by their ability to use dynamic, input-dependent state transition matrices.

As suggested, we have added a new paragraph to the Discussion section to address this while also proposing a viable path forward that can enable dynamic transition matrix implementation:

In p16

‘Our current implementation assumes static, non-input-dependent state transition matrix, a design choice mirroring early SSMs like S4 and S5 that prioritizes straightforward mapping onto weight-stationary CIM hardware. While highly efficient and friendly to the CIM architecture, this static approach prevents the model from dynamically adjusting its parameters based on the input content which is a key capability behind the superior performance of modern selective SSMs such as Mamba. However, as recent device-level research demonstrates, the time constant of the SN can be actively modulated in-situ by an external control signal, such as a bias voltage⁶⁶. Such approaches make it possible to dynamically adjust the state decay rates on-the-fly and effectively implement modern selective SSMs in the proposed architecture.’

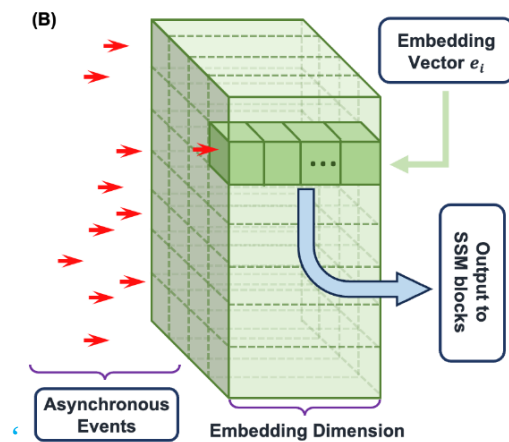
66. Li, Y. et al. Homogeneous Memristors with Tunable Decay Dynamics for Self-Adaptive Reservoir Computing. *ACS Nano* **19**, 28186–28196 (2025).

Results

- The graph in Figure 1(B) is unclear to me (the square).

Response:

We have revised Figure 1(B) in the new manuscript. The vague "square" has been replaced with a detailed diagram illustrating the physical implementation of the channel-wise embedding layer. The revised Fig. 1B is shown below, along with updated Figure captions:



(B) Functional diagram of the channel-wise embedding layer. An incoming event activates the embedding layer which produces a D -dimensional vector depending on the event's channel index.

- The discussion on nonlinearities could use the phrase "positional nonlinearity" which 1) preserves universal function approximation, 2) without breaking the ability to represent training as 1D-conv.

Response:

We thank the reviewer for this excellent and precise suggestion. As suggested, we have revised the manuscript to adopt this terminology. We now refer to these operations as "positional non-linear function" and "positional non-linear gating" throughout the text.

- In model training, it's said that BPTT is used. Training an SSM is indeed functionally equivalent to BPTT, though some folk take issue with it as BPTT is defined as having linear time complexity while training SSMs follows log-complexity with time. I don't

think there is concrete guidance here - I have no fixation on this if you prefer not to change this.

Response:

Thank you for the excellent comment. To clarify this point, we have added a new subsection in Methods titled "Parallelized BPTT". In this new subsection, we explicitly state that we do not compute the recurrence sequentially:

In p18

‘Parallelized BPTT

We train the model using backpropagation through time, computing gradients with respect to all temporal dependencies via automatic differentiation. The forward pass implements the discrete-time SSM recurrence $h(t + \Delta t) = \bar{A}(\Delta t)h(t) + \bar{B}x(t + \Delta t)$ for sequences of length L . Rather than computing this recurrence sequentially, we leverage a parallel associative scan algorithm that exploits the associativity of linear recurrences through a operator $(A_i, b_i) \circ (A_j, b_j) = (A_j \cdot A_i, A_j \cdot b_i + b_j)$. This formulation enables simultaneous computation of all hidden states via a balanced binary tree reduction, achieving $O(\log L)$ parallel time complexity compared to $O(L)$ for sequential evaluation given sufficient parallel processors⁷⁴. During the backward pass, automatic differentiation through the scan operation computes gradients for all SSM parameters—including state matrix A , input projection B , output projection C , and learnable discretization timesteps—while preserving the same logarithmic complexity. This parallelized BPTT implementation allows efficient gradient-based optimization on long event sequences while maintaining full temporal credit assignment across all time steps.’

74. Blelloch, G. E. Prefix sums and their applications. in 1294199 Bytes (Carnegie Mellon University, 2004). doi:10.1184/R1/6608579.V1.

- It is unclear to me where the asynchronosity in the SSM comes from. What is sparsifying the activations?

Response:

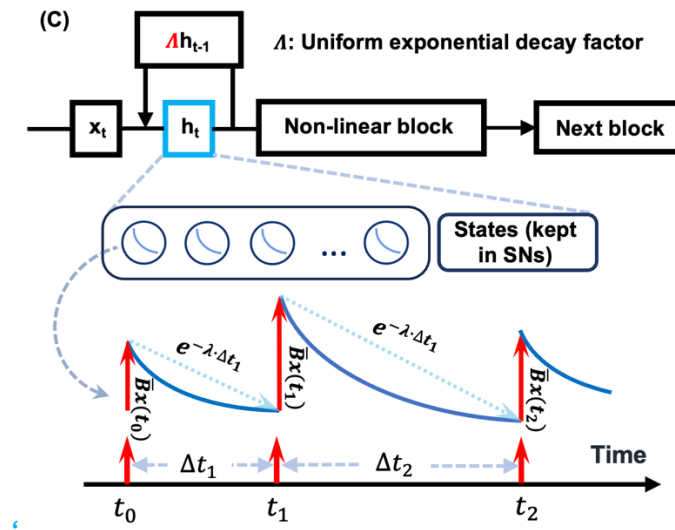
We thank the reviewer for this good question, which allows us to clarify a key aspect of our design. The asynchronicity in our system comes from two primary sources: (1) the

event-driven nature of the input data and (2) the clockless physical implementation of the SSM state decay. We have added a new statement after the we introduce the architecture:

In p5

‘As illustrated in Fig. 1(C), When no inputs are applied (e.g., during intervals Δt_1 and Δt_2), the state evolve passively and natively through the short-term memory property of the SNs, implementing the exponential decay (e.g., $e^{-\Delta t_1}, e^{-\Delta t_2}$) without the need of clocks or synchronization. Active computation is needed only when an input arrives (e.g., at time t_1, t_2), where the projected input term (e.g., $\bar{B}x(t_0), \bar{B}x(t_1), \bar{B}x(t_2)$) is computed via VMM and added to the current state, making the system operation fully event-driven. The updated state is then read out, further processed through a non-linear block, and applied as an input event to trigger the next block's computation. By leveraging physics, including Ohm's law, Kirchhoff's current law, and internal device dynamics, this approach achieves highly parallel and energy-efficient computation and is compatible with asynchronous inputs.’

Fig. 1(C) is attached:



Operational schematic of a single SSM block (upper). The state of the SSM is represented by those of individual state nodes (SNs) (middle) in the proposed implementation. The state in an individual SN decays exponentially following a trained decay time constant. When an input x arrives, it is projected onto the state space through $\bar{B}x$ and is added to the current state. The updated state h is then read out, passed through some non-linear functions and sent to the next block. Note actions (i.e. computations) are only performed when an event arrives, allowing asynchronous operations.’

Regarding the reviewer's second question, "What is sparsifying the activations?" We have added a new paragraph in Discussion section:

In p16

'Unlike SNNs, which often leverage both input sparsity and internal activation sparsity (i.e., neurons only compute when they spike), our current SSM implementation does not enforce sparse internal activations. The system's efficiency is primarily derived from leveraging the input sparsity of the event-based data; active computation (e.g., VMM) is needed only when an event arrives, while the state evolution is handled passively by device physics. Inducing sparsity in the internal SSM activations, perhaps through thresholding or other mechanisms, could offer a path to even greater energy savings and remains a direction for future research.'

- While not essential, an ablation study to identify what features of the RRAM/ADC non-idealities impact performance.

Response:

Thank you for the valuable comment. The main non-idealities brought by the RRAM/ADC are quantization and inherent variations. In the paper, we study these two effects step by step. One is that we use QAT to check the effect of the quantization, and the other is that we inject the random noise that accounts device variation effects during computing. Additionally, we now also include DVS-Gesture dataset results in these non-idealities studies:

In p14

'We next evaluate the model implementation on CIM hardware using the SSC and DVS-Gesture datasets. To align the SSM model with analog CIM hardware constraints, all weights and activations are quantized to 8-bit integer (INT8) using quantization-aware training (QAT)^{64,65}. This step causes no accuracy degradation for the DVS-Gesture model (remaining at 97.3%) and only a minimal 0.3% drop for the SSC model (from 84.7% to 84.4%).

We then studied the effect of device non-idealities in analog hardware by introducing random hardware variations to the INT8 parameters (see Method) based on the measured device variation effect data in Fig. 2(B-D). This resulted in only a modest accuracy decline for both models without any re-training or other mitigation techniques: the SSC accuracy dropped 2.39% to 82.01%, and the DVS-Gesture accuracy dropped 2.2% to 95.1%. These results support the resilience of the proposed model for practical analog

hardware implementations, likely due to the use of only fully-connect layers and the shallow model architecture. Future implementation of techniques such as variation-aware-training⁶⁵ is expected to further improve the hardware accuracy.'

64. Jacob, B. et al. Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. in 2704–2713 (2018).

65. Wang, Q., Park, Y. & Lu, W. D. Device Variation Effects on Neural Network Inference Accuracy in Analog In-Memory Computing Systems. *Advanced Intelligent Systems* **4**, 2100199 (2022).

- There is limited information on how sparsity provides an advantage in this particular set up.

Response:

We thank the reviewer for raising this important point. To address this, we have added a new computational complexity analysis to the "Model Training and Performance Analysis" section (with full details in Methods under "Theoretical FLOPs Analysis"). This analysis directly quantifies the benefits of processing sparse event data:

In p12

'To provide a quantitative basis for the hardware and latency advantages of our approach, we performed a detailed computational complexity analysis (see Methods). For the DVS128 Gesture dataset, our Event-SSM requires 1.68 GFLOPs to process a typical data sequence. In contrast, a common alternative approach involves converting the event stream into video frames (e.g., at 30fps) and using a standard CNN. A ResNet-18 based model processing these frames would require 104.28 GFLOPs (a 62-fold increase), and a ResNet-50 would require 219.80 GFLOPs (a 131-fold increase). This significant reduction in computational load stems from our model's ability to leverage the inherent sparsity of input event data, avoiding the massive computational overhead associated with frame-based processing.'

In Method section

'Theoretical FLOPs Analysis

We conducted a theoretical FLOPs analysis to compare the computational complexity of Event-SSM and ResNet models for DVS-Gesture data processing. The Event-SSM model has 2 stages each composing of 3 SSM blocks. The state in the first stage has the dimension

of 128 while the second has 256. We assume the input has sequences of 65,536 events which is typical to DVS-Gesture dataset.

For models based on synchronous processing, event data are converted to 30fps video (approximately 180 frames for 6 seconds), with each frame having a resolution of $128 \times 128 \times 2$ channels (ON/OFF events). We analyzed both ResNet-18 and ResNet-50 as standard architectures.

Our analysis assumes: (1) multiply-accumulate (MAC) operations count as 2 FLOPs, (2) matrix multiplication $H_{in} \times H \times H_{out}$ counts as $2 \times H_{in} \times H \times H_{out}$ FLOPs, and (3) normalization operations count as 5 FLOPs per element. (4) ResNet processes each frame independently.

For Event-SSM, we calculate FLOPs for each layer including embedding projection ($L \times D \times 2$), and for each SSM layer: input projection ($L \times H_{in} \times H \times 2$), state evolution ($L \times H_{in} \times 2$), output projection ($L \times H \times H_{out} \times 2$), feedforward network ($L \times H_{out}^2 \times 2$), and normalization ($L \times H \times 5$), where D , L , H_{in} , H , H_{out} denote embedding dimension, sequence length, input feature dimension, and state dimension, output feature dimension respectively. Pooling with stride 16 reduces the sequence length from 65,536 to 4,096 after the first stage, and to 256 after the second stage. For ResNet, we use the standard formula $FLOPs = H_{out} \times W_{out} \times C_{out} \times (K^2 \times C_{in} \times 2)$ for each convolutional layer, where H_{out} , W_{out} , C_{out} , K , C_{in} denote the output height, output width, number of output channels, kernel size, and number of input channels, respectively. Total FLOPs equal per-frame FLOPs multiplied by the number of frames (180)

Results show that Event-SSM requires 1.68 GFLOPs, while ResNet-18 and ResNet-50 require 104.28 and 219.80 GFLOPs, respectively. This corresponds to $62.12 \times$ and $130.93 \times$ computational efficiency improvements for Event-SSM, primarily attributed to (1) the sparsity of event representation (65,536 events vs. 5,898,240 pixel values, a $90 \times$ density difference), (2) elimination of fixed frame-rate processing overhead, (3) compact model design, and (4) real-valued operations throughout.'

Reviewer #3:

This manuscript presents an algorithm-hardware co-design approach for efficient, real-time, and asynchronous processing using state space models (SSMs). The key contributions can be summarized as follows: (1) a re-parameterized 6-block SSM model in which the decay parameter (λ) is shared within each block, thereby reducing hardware complexity while improving accuracy; (2) a memristor-based compute-in-memory (CIM) architecture to accelerate input vector-matrix multiplications (VMMs); and (3) an exploitation of the intrinsic short-term conductance decay behavior of memristors to perform hidden state (h) updates. However, several parts of the manuscript lack clarity and require further elaboration and justification.

Response:

Thank you for your valuable comments and suggestions.

1. Recent References on RRAM CIM

The reference list regarding RRAM CIM could be strengthened by including recent works published in Nature and Nature Electronics. Incorporating these high-impact studies would better position the manuscript within the current state of the art and highlight its relevance to ongoing developments.

Response:

We thank the reviewer for this valuable suggestion. To better position our manuscript within the current state-of-the-art, we have updated our Introduction and Reference sections to include several recent high-impact studies on RRAM-based Compute-in-Memory from Nature, Nature Electronics, and Nature Computational Science. These additions help highlight the timeliness and relevance of our work to ongoing developments in the field:

In p2

‘Furthermore, from a hardware perspective, convolution and transformer blocks used in the SNN models are not well suited for energy efficient architectures such as compute-in-memory (CIM). In CIM, weights are stored stationarily in high-density memory cells such as Resistive RAM (RRAM)^{17–19} or Phase-Change Memory (PCM)^{20,21}, and computations, primarily vector-matrix multiplication (VMM), are directly performed inside the memory array through Ohm’s law and Kirchhoff’s current law^{22,23}. This in-situ computation can dramatically minimize data movement compared to traditional von Neumann designs^{24–26}. However, for convolution operations, the same weights are re-used for many input patches

thus the weight read costs can be efficiently amortized in well-designed digital systems, diminishing the benefits of CIM and amplifying the challenges of the slower read in emerging devices used for CIM implementations. The need to load different input patches from the feature map memory to perform convolutions also requires data re-shuffling, again diminishing CIM benefits. As a result, convolution operations are generally not well matched with CIM implementations. The attention mechanism, representing input-dependent weights, is also incompatible with stationary CIM arrays. Additionally, the use of Key-Value (KV) cache which increases quadratically with input sequence length and needs to be constantly updated also necessitates hybrid designs that delegate these operations to separate digital circuitry²⁷.

In p17

‘Finally, in this study we prioritized computational fidelity and robustness, including the use of differential positive and negative weight arrays and 1T1R cells in the CIM chip. More advanced techniques such as 2’s complement¹⁹ and passive crossbar arrays⁷¹ can potentially also be used, offering benefits of higher compute density. Fully analog implementations that eliminate ADCs and DACs can also further improve the system’s energy efficiency, achieving clockless asynchronous signal processing.’

17. Lanza, M. et al. Memristive technologies for data storage, computation, encryption, and radio-frequency communication. *Science* **376**, eabj9979 (2022).
18. Huo, Q. et al. A computing-in-memory macro based on three-dimensional resistive random-access memory. *Nat Electron* **5**, 469–477 (2022).
19. Khwa, W.-S. et al. A mixed-precision memristor and SRAM compute-in-memory AI processor. *Nature* **639**, 617–623 (2025).
20. Wong, H.-S. P. et al. Phase Change Memory. *Proceedings of the IEEE* **98**, 2201–2227 (2010).
21. Le Gallo, M. et al. A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference. *Nat Electron* **6**, 680–693 (2023).
22. Yang, J. J., Strukov, D. B. & Stewart, D. R. Memristive devices for computing. *Nature Nanotech* **8**, 13–24 (2013).
23. Shafiee, A. et al. ISAAC: a convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *SIGARCH Comput. Archit. News* **44**, 14–26 (2016).

24. Wang, X. *et al.* TAICHI: A Tiled Architecture for In-Memory Computing and Heterogeneous Integration. *IEEE Transactions on Circuits and Systems II: Express Briefs* **69**, 559–563 (2022).
25. Ambrogio, S. *et al.* An analog-AI chip for energy-efficient speech recognition and transcription. *Nature* **620**, 768–775 (2023).
26. Wang, Z. *et al.* A dual-domain compute-in-memory system for general neural network inference. *Nat Electron* **8**, 276–287 (2025).
27. Büchel, J. *et al.* Efficient scaling of large language models with mixture of experts and 3D analog in-memory computing. *Nat Comput Sci* **5**, 13–26 (2025).
71. Choi, S., Bezugam, S. S., Bhattacharya, T., Kwon, D. & Strukov, D. B. Wafer-scale fabrication of memristive passive crossbar circuits for brain-scale neuromorphic computing. *Nat Commun* **16**, 8757 (2025).

The new included references are:

18. Huo, Q. *et al.* A computing-in-memory macro based on three-dimensional resistive random-access memory. *Nat Electron* **5**, 469–477 (2022).
19. Khwa, W.-S. *et al.* A mixed-precision memristor and SRAM compute-in-memory AI processor. *Nature* **639**, 617–623 (2025).
21. Le Gallo, M. *et al.* A 64-core mixed-signal in-memory compute chip based on phase-change memory for deep neural network inference. *Nat Electron* **6**, 680–693 (2023).
26. Wang, Z. *et al.* A dual-domain compute-in-memory system for general neural network inference. *Nat Electron* **8**, 276–287 (2025).
27. Büchel, J. *et al.* Efficient scaling of large language models with mixture of experts and 3D analog in-memory computing. *Nat Comput Sci* **5**, 13–26 (2025).
71. Choi, S., Bezugam, S. S., Bhattacharya, T., Kwon, D. & Strukov, D. B. Wafer-scale fabrication of memristive passive crossbar circuits for brain-scale neuromorphic computing. *Nat Commun* **16**, 8757 (2025).

2. Crossbar vs. 1T1R Structures

The manuscript should address the impact of sneak current in crossbar arrays and

provide a discussion comparing the advantages and trade-offs of crossbar versus 1T1R structures. Such an analysis would help readers better appreciate the design choices and their implications for scalability and reliability.

Response:

We thank the reviewer for raising this point. In this study we prioritized computational fidelity and robustness, including the use of differential positive and negative weight arrays and 1T1R cells in the CIM chip. We agree that more advanced techniques such as 2's complement and passive crossbar arrays can potentially also be used, offering benefits of higher compute density. We have added these discussions to the Discussion section:

In p17

'Finally, in this study we prioritized computational fidelity and robustness, including the use of differential positive and negative weight arrays and 1T1R cells in the CIM chip. More advanced techniques such as 2's complement¹⁹ and passive crossbar arrays⁷¹ can potentially also be used, offering benefits of higher compute density. Fully analog implementations that eliminate ADCs and DACs can also further improve the system's energy efficiency, achieving clockless asynchronous signal processing.'

19. Khwa, W.-S. et al. A mixed-precision memristor and SRAM compute-in-memory AI processor. *Nature* **639**, 617–623 (2025).

71. Choi, S., Bezugam, S. S., Bhattacharya, T., Kwon, D. & Strukov, D. B. Wafer-scale fabrication of memristive passive crossbar circuits for brain-scale neuromorphic computing. *Nat Commun* **16**, 8757 (2025).

3. Figure 1(A) & (B): Clarity of Asynchronous Event Representation

The presentation of asynchronous events through the channel-wise embedding layer is unclear. For instance, the y-axis values are not specified, and it is difficult to determine whether the spikes in (A) correspond directly to the square wave in (B). Providing clarification or additional explanation would greatly improve readability.

Response:

Thank you for this comment. We have substantially revised Figure 1 and its accompanying description in the manuscript to address this unambiguity. Copied below:

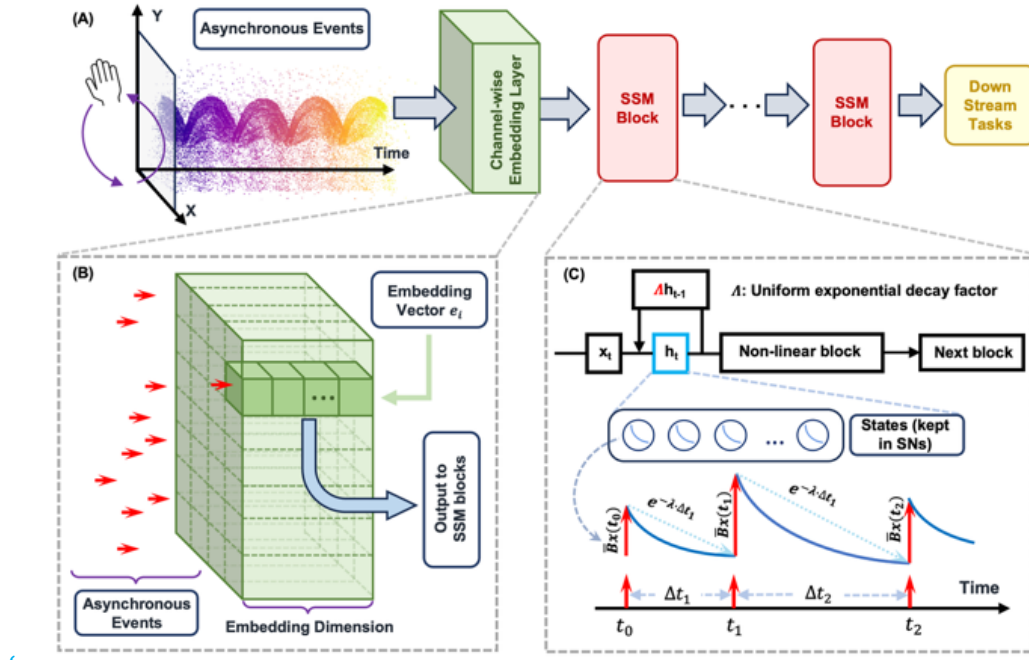


Figure 1. Model architecture for the proposed event-driven SSM. (A) High-level network architecture, including a channel-wise embedding layer that maps incoming events to a high-dimensional latent state space and stacked SSM blocks that asynchronously process information in the state space; (B) Functional diagram of the channel-wise embedding layer. An incoming spike activates the embedding layer which produces a D-dimensional vector depending on the event's channel index. (C) Operational schematic of a single SSM block (upper). The state of the SSM is represented by those of individual state nodes (SNs) (middle) in the proposed implementation. The state in an individual SN decays exponentially following a trained decay time constant. When an input x arrives, it is projected onto the state space through $B\bar{x}$ and is added to the current state. The updated state h is then read out, passed through some non-linear functions and sent to the next block. Note actions (i.e. computations) are only performed when an event arrives, allowing asynchronous operations.'

4. Synchronous vs. Asynchronous Processing

The manuscript emphasizes the importance of asynchronous processing, but the distinction from synchronous processing is not sufficiently detailed. The authors are encouraged to elaborate on the differences, including their respective advantages and disadvantages. For example, in Table 2, Peng et al. [50] report comparable

performance despite not being asynchronous. Enhancing this comparison table to highlight the significance of the proposed approach would strengthen the manuscript.

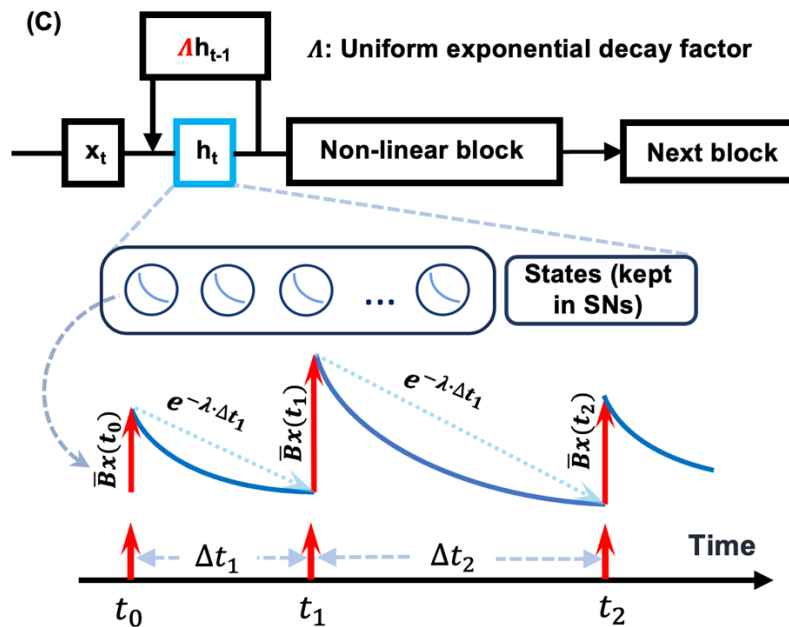
Response:

Thank you for this valuable comment. We have revised the text in the Introduction section and Discussion section to explain how asynchronous processing is achieved in the proposed system:

In p5

‘As illustrated in Fig. 1(C), When no inputs are applied (e.g., during intervals Δt_1 and Δt_2), the state evolve passively and natively through the short-term memory property of the SNs, implementing the exponential decay (e.g., $e^{-\Delta t_1}, e^{-\Delta t_2}$) without the need of clocks or synchronization. Active computation is needed only when an input arrives (e.g., at time t_1, t_2), where the projected input term (e.g., $\bar{B}x(t_0), \bar{B}x(t_1), \bar{B}x(t_2)$) is computed via VMM and added to the current state, making the system operation fully event-driven. The updated state is then read out, further processed through a non-linear block, and applied as an input event to trigger the next block's computation. By leveraging physics, including Ohm's law, Kirchhoff's current law, and internal device dynamics, this approach achieves highly parallel and energy-efficient computation and is compatible with asynchronous inputs.’

Fig. 1(C) is also attached below:



(C) Operational schematic of a single SSM block (upper). The state of the SSM is represented by those of individual state nodes (SNs) (middle) in the proposed implementation. The state in an individual SN decays exponentially following a trained decay time constant. When an input x arrives, it is projected onto the state space through $B^T x$ and is added to the current state. The updated state h is then read out, passed through some non-linear functions and sent to the next block. Note actions (i.e. computations) are only performed when an event arrives, allowing asynchronous operations.'

Regarding other approaches such as Peng et al. [50], we note Peng et al. [50] used Transformer as the backbone, which we have discussed earlier to be not compatible with CIM. Additionally, Pent et al. [50] used convolutions including group convolutions and group aggregation which are frame-based techniques. We have added a discussion paragraph in the manuscript, immediately following the presentation of Table 2, to address how the proposed approach differs from frame-based approaches. Copied below:

In p12

'It is noteworthy that some synchronous methods, by aggregating sparse events into frame-like structures, can achieve higher accuracy on DVS dataset. However, this performance comes at the cost of forgoing the benefits of event-based sensing. The frame-creation step not only introduces additional latency but also increases the data volume, leading to more computational demands. In contrast, the proposed asynchronous SSM processes each event as it arrives, preserving the data's inherent sparsity and temporal precision. This makes our approach suited for real-time, resource-constrained applications where minimizing latency and power consumption is paramount.'

To provide a much more direct and convincing case, we have added a detailed computational complexity analysis (in FLOPs) to the manuscript. This analysis compares our Event-SSM to ResNet models for processing the DVS128 Gesture dataset (full details are in the Methods section under "Theoretical FLOPs Analysis"):

In p12

'To provide a quantitative basis for the hardware and latency advantages of our approach, we performed a detailed computational complexity analysis (see Methods). For the DVS128 Gesture dataset, our Event-SSM requires 1.68 GFLOPs to process a typical data sequence. In contrast, a common alternative approach involves converting the event stream into video frames (e.g., at 30fps) and using a standard CNN. A ResNet-18 based model processing these frames would require 104.28 GFLOPs (a 62-fold increase), and a ResNet-50 would require 219.80 GFLOPs (a 131-fold increase). This significant

reduction in computational load stems from our model's ability to leverage the inherent sparsity of input event data, avoiding the massive computational overhead associated with frame-based processing.'

In Method section

'Theoretical FLOPs Analysis

We conducted a theoretical FLOPs analysis to compare the computational complexity of Event-SSM and ResNet models for DVS-Gesture data processing. The Event-SSM model has 2 stages each composing of 3 SSM blocks. The state in the first stage has the dimension of 128 while the second has 256. We assume the input has sequences of 65,536 events which is typical to DVS-Gesture dataset.

For models based on synchronous processing, event data are converted to 30fps video (approximately 180 frames for 6 seconds), with each frame having a resolution of $128 \times 128 \times 2$ channels (ON/OFF events). We analyzed both ResNet-18 and ResNet-50 as standard architectures.

Our analysis assumes: (1) multiply-accumulate (MAC) operations count as 2 FLOPs, (2) matrix multiplication $H_{in} \times H \times H_{out}$ counts as $2 \times H_{in} \times H \times H_{out}$ FLOPs, and (3) normalization operations count as 5 FLOPs per element. (4) ResNet processes each frame independently.

For Event-SSM, we calculate FLOPs for each layer including embedding projection ($L \times D \times 2$), and for each SSM layer: input projection ($L \times H_{in} \times H \times 2$), state evolution ($L \times H_{in} \times 2$), output projection ($L \times H \times H_{out} \times 2$), feedforward network ($L \times H_{out}^2 \times 2$), and normalization ($L \times H \times 5$), where D , L , H_{in} , H , H_{out} denote embedding dimension, sequence length, input feature dimension, and state dimension, output feature dimension respectively. Pooling with stride 16 reduces the sequence length from 65,536 to 4,096 after the first stage, and to 256 after the second stage. For ResNet, we use the standard formula $FLOPs = H_{out} \times W_{out} \times C_{out} \times (K^2 \times C_{in} \times 2)$ for each convolutional layer, where H_{out} , W_{out} , C_{out} , K , C_{in} denote the output height, output width, number of output channels, kernel size, and number of input channels, respectively. Total FLOPs equal per-frame FLOPs multiplied by the number of frames (180)

Results show that Event-SSM requires 1.68 GFLOPs, while ResNet-18 and ResNet-50 require 104.28 and 219.80 GFLOPs, respectively. This corresponds to $62.12 \times$ and $130.93 \times$ computational efficiency improvements for Event-SSM, primarily attributed to

(1) the sparsity of event representation (65,536 events vs. 5,898,240 pixel values, a $90 \times$ density difference), (2) elimination of fixed frame-rate processing overhead, (3) compact model design, and (4) real-valued operations throughout'

5. Figure 2(A–D): Caption and Interpretation

The caption states “Analysis of model architecture and decay parameter (λ) choices on classification accuracy,” yet the figure appears to show the number of blocks versus accuracy. Does the caption misstate the content, or is there a specific relationship between λ and the number of blocks? Furthermore, the authors claim that the proposed model requires only six blocks, whereas counterparts such as ResNet require more than 30 layers. Given the distinction between the terms “block” and “layer,” is it valid to conclude that the proposed model achieves lower latency? A direct comparison in terms of latency or other performance metrics would make this claim more convincing.

Response:

Thank you for pointing out these ambiguities. We have revised Figure 2 and its captions (now Figure 3) to address the points you have mentioned:

Regarding the Figure Caption (now Figure 3): We have revised the caption to more accurately reflect the entire figure's content. The new caption is "*Effect of model depth and decay parameter (λ) configuration on classification accuracy.*" This caption now correctly covers Fig. 3(A–D), which show the impact of model depth (number of sequential SSM blocks) on accuracy. It also covers Fig. 3 (E) and (F), which are dedicated to analyzing the decay parameter (λ) configuration. We have also standardized the terminology to "SSM block" throughout the manuscript for clarity

Regarding the latency claim, indeed the proposed system requires much fewer FLOPS and should lead to improved latency. We have added a detailed computational complexity analysis, which was included in response to your Comment 4, also copied below:

In Method section

'Theoretical FLOPs Analysis

We conducted a theoretical FLOPs analysis to compare the computational complexity of Event-SSM and ResNet models for DVS-Gesture data processing. The Event-SSM model, has 2 stages, and the state in the first stage has the dimension of 128 while the second has

256. Each state composes of 3 SSM blocks. We assume the sample has the sequences of 65,536 events which is typical to DVS-Gesture dataset.

Event data is converted to 30fps video (approximately 180 frames for 6 seconds), with each frame having a resolution of $128 \times 128 \times 2$ channels (ON/OFF events). We analyzed both ResNet-18 and ResNet-50 standard architectures.

Our analysis assumes: (1) multiply-accumulate (MAC) operations count as 2 FLOPs, (2) matrix multiplication $H_{in} \times H \times H_{out}$ counts as $2 \times H_{in} \times H \times H_{out}$ FLOPs, and (3) normalization operations count as 5 FLOPs per element. (4) ResNet processes each frame independently.

For Event-SSM, we calculate FLOPs for each layer including embedding projection ($L \times D \times 2$), and for each SSM layer: input projection ($L \times H_{in} \times H \times 2$), state evolution ($L \times H_{in} \times 2$), output projection ($L \times H \times H_{out} \times 2$), feedforward network ($L \times H_{out}^2 \times 2$), and normalization ($L \times H \times 5$), where D , L , H_{in} , H , H_{out} denote embedding dimension, sequence length, input feature dimension, and state dimension, output feature dimension respectively. Pooling with stride 16 reduces the sequence length from 65,536 to 4,096 after the first stage, and to 256 after the second stage. For ResNet, we use the standard formula $FLOPs = H_{out} \times W_{out} \times C_{out} \times (K^2 \times C_{in} \times 2)$ for each convolutional layer, where H_{out} , W_{out} , C_{out} , K , C_{in} denote the output height, output width, number of output channels, kernel size, and number of input channels, respectively. Total FLOPs equals per-frame FLOPs multiplied by the number of frames (180)

Results show that Event-SSM requires 1.68 GFLOPs, while ResNet-18 and ResNet-50 require 104.28 and 219.80 GFLOPs, respectively. This corresponds to $62.12 \times$ and $130.93 \times$ computational efficiency improvements for Event-SSM, primarily attributed to (1) the sparsity of event representation (65,536 events vs. 5,898,240 pixel values, a $90 \times$ density difference), (2) elimination of fixed frame-rate processing overhead, (3) compact model design, and (4) real-valued operations throughout.'

6. RRAM Mapping Strategy

The authors map positive and negative weights onto separate RRAM arrays, while prior work often employs 2's complement mapping, requiring only a single array. What motivated the use of separate arrays in this work? Discussion of the resulting area overhead and associated trade-offs would add valuable context.

Response:

Thank you for this valuable comment. As explained in response to your Comment 2 in this study we prioritized computational fidelity and robustness, including the use of differential positive and negative weight arrays and 1T1R cells in the CIM chip. We agree

that passive crossbar arrays can potentially also be used and will offer benefits of higher compute density. We have added these discussions to the Discussion section:

In p17

‘Finally, in this study we prioritized computational fidelity and robustness, including the use of differential positive and negative weight arrays and 1T1R cells in the CIM chip. More advanced techniques such as 2’s complement¹⁹ and passive crossbar arrays⁷¹ can potentially also be used, offering benefits of higher compute density. Fully analog implementations that eliminate ADCs and DACs can also further improve the system’s energy efficiency, achieving clockless asynchronous signal processing.’

19. Khwa, W.-S. et al. A mixed-precision memristor and SRAM compute-in-memory AI processor. *Nature* **639**, 617–623 (2025).

71. Choi, S., Bezugam, S. S., Bhattacharya, T., Kwon, D. & Strukov, D. B. Wafer-scale fabrication of memristive passive crossbar circuits for brain-scale neuromorphic computing. *Nat Commun* **16**, 8757 (2025).

7. Fixed Decay Parameter (λ) and Accuracy

Line 276 states that inference accuracy improves when fixing the decay parameter (λ). The authors should provide a more detailed mathematical derivation or theoretical explanation to clarify why and how fixing λ enhances accuracy.

Response:

We thank the reviewer for this insightful question. This is indeed an interesting and non-obvious result that we observed.

While a formal mathematical derivation explaining this phenomenon is still ongoing, our analysis strongly suggests that fixing λ acts as an effective regularization technique, which stabilizes the training process for these event-driven datasets. Our evidence for this is most clearly demonstrated by the Spiking Speech Commands (SSC) dataset. As shown in our results, when λ was left as a freely learnable parameter, the training process became unstable and diverged, resulting in "Not a Number" (NaN) loss values. Our analysis in Figure 3(F) revealed that just before this divergence, the learned decay time constants grew excessively large. This slow decay causes the hidden states to accumulate without being forgotten, leading to exploding gradients and numerical instability. By fixing λ to a layer-wise average (determined from an initial training

phase), we effectively constrain the optimization landscape. This constraint prevents the model from exploring these unstable regions, forcing it to find a more robust and stable solution. This regularization effect not only enabled the SSC model to converge successfully (achieving 84.7% accuracy) but also led to modest accuracy improvements on the other datasets.

We agree that a deeper theoretical investigation into how this constraint acts as a regularizer is a fascinating and valuable direction for future research. Additionally, we also acknowledge that recent device level research has shown that decay constants in STM memristors can be adjusted in-situ, potentially eliminating the need for fixed λ (at a cost of additional circuitry). We have added a paragraph discussing this point in the revised manuscript, copied below:

In p16

‘Our current implementation assumes static, non-input-dependent state transition matrix, a design choice mirroring early SSMs like S4 and S5 that prioritizes straightforward mapping onto weight-stationary CIM hardware. While highly efficient and friendly to the CIM architecture, this static approach prevents the model from dynamically adjusting its parameters based on the input content which is a key capability behind the superior performance of modern selective SSMs such as Mamba. However, as recent device-level research demonstrates, the time constant of the SN can be actively modulated in-situ by an external control signal, such as a bias voltage⁶⁶. Such approaches make it possible to dynamically adjust the state decay rates on-the-fly and effectively implement modern selective SSMs in the proposed architecture.’

66. Li, Y. et al. Homogeneous Memristors with Tunable Decay Dynamics for Self-Adaptive Reservoir Computing. *ACS Nano* **19**, 28186–28196 (2025)

8. Flexibility Limitations of Memristor Conductance Decay

Using memristor conductance decay as RN is an intriguing approach. However, because the decay rate is largely determined by oxide thickness during fabrication, flexibility may be limited for supporting different SSM models or enabling weight updates. The authors should elaborate on potential solutions to overcome this limitation, particularly given the importance of reprogramming capabilities at the edge.

Response:

Thank you for highlighting this important point. As discussed in response to your Comment 7, recent device research has shown that if needed, it is possible to adjust

the decay time constant in STM memristors on the fly. We have added a paragraph in Discussion section about the reprogrammable decay as a potential future direction.

9. Voltage Levels and Endurance in Conductance Updates

In Equation (9), the horizontal line voltages of the input projection memristor array are used as input spikes $x(t+\Delta t)$ to update conductance. The authors should clarify the voltage levels applied in this process. Moreover, since each conductance update effectively involves multiple write operations, a discussion of endurance concerns under repeated updates would be valuable.

Response:

We thank the reviewer for these important questions. We provide the following clarifications on the voltage levels and endurance considerations.

Voltage Levels for Conductance Updates

The conductance update is performed by applying a burst of **N 1.4 V / 50 μ s pulses**. The number of pulses **N is proportional to the VMM output**.

We have revised the manuscript (in the Method) to explicitly state these values.

In p20

‘For each SN updates, which corresponds to an input spike $x(t+\Delta t)$ in Eq. 8, a burst of N 1.4 V / 50 μ s pulses were applied on the memristor, where the number of pulses N is proportional to the VMM output.’

Endurance Considerations

We agree that endurance is an important topic. We have added a new paragraph to the Discussion section to address device endurance under repeated, incremental updates, and possible implementations that offer even higher endurance.

In p16-17

‘Since the SN state is updated when an event arrives, the endurance of the SN devices needs to be high enough as each update constitutes a write operation. It is important to note that these updates are small and incremental which is generally less stressful for the device, rather than full SET/RESET cycles. The WOX memristors used in our work have demonstrated programming endurance for more than 10^8 cycles^{67,68}. Additionally, other STM memristors such as those based on metal-insulator transition (NbOx)⁶⁹, or spin-transfer torque devices⁷⁰, have shown even longer endurance and can potentially be used in the proposed architecture as well.’

67. Kim, K.-H., Hyun Jo, S., Gaba, S. & Lu, W. Nanoscale resistive memory with intrinsic diode characteristics and long endurance. *Appl. Phys. Lett.* 96, 053106 (2010).
68. Lai, E.-K. et al. Tungsten Oxide Resistive Memory Using Rapid Thermal Oxidation of Tungsten Plugs. *Jpn. J. Appl. Phys.* 49, 04DD17 (2010).
69. Kumar, S., Strachan, J. P. & Williams, R. S. Chaotic dynamics in nanoscale NbO₂ Mott memristors for analogue computing. *Nature* 548, 318–321 (2017).
70. Grollier, J. et al. Neuromorphic spintronics. *Nat Electron* 3, 360–370 (2020).

Response Letter

Title:

Compute-in-Memory Implementation of State Space Models for Event Sequence Processing

We thank all reviewers for their valuable and constructive comments on this paper. We are pleased to learn that all reviewers agreed our revisions addressed their previous comments. In the revised manuscript we addressed the additional valuable suggestions and comments by the reviewers. The detailed point-by-point response can be found below.

Reviewer #1:

Comments to the Author:

I thank the authors for having carefully taken my comments into account. The authors made substantial improvements on the manuscript. The claimed points are now endorsed by enough details and results.

My only two minor comments are:

1. Please avoid using some strange wordings like 'software-level'. Do you mean simulation? If so please simply use 'simulation'.

Response:

We thank reviewer for the positive feedback and the valuable comments.

Following the reviewer's suggestion, we have changed the text correspondingly:

In p8

'We note that the training process is conducted in simulation.'

In p10:

'Table 1 summarizes the simulation results on two prominent spiking audio datasets, which are frequently tackled with SNN-based architectures.'

In p13:

'We performed several additional analyses in simulation to analyze the model behavior.'

And the caption of the Table 1 and Table 2 are also changed correspondingly:

'Table 1: Comparison of our SSM model to the state-of-the-art SNN based models on Spike Audio Datasets in simulation'

'Table 2: Comparison of our SSM model to other leading models on the DVS128 Gesture and DVS128 Lips datasets in simulation.'

2. Now I understand that the main difference between the proposed method and reservoir computing is that the decay factor λ of the devices were also trained, providing extra flexibilities for the system to better adapt to the tasks. My only concern is that was the λ trained offline and then somehow configured

the devices to the values of target lambda? Please clarify. If so, how to overcome the device variabilities and sim2real problems?

Response:

Thank you for this question. Your understanding is correct: λ is determined during offline training, and we then implement these target values through physical STM memristor devices.

To address the concern about variations in the STM memristor devices, we conducted a new simulation study to evaluate the model's robustness against variations in λ . This result is included in the revised manuscript as Supplementary Information S5. Copied below:

'S5. Model Performance against Decay Constant (λ) Variations.

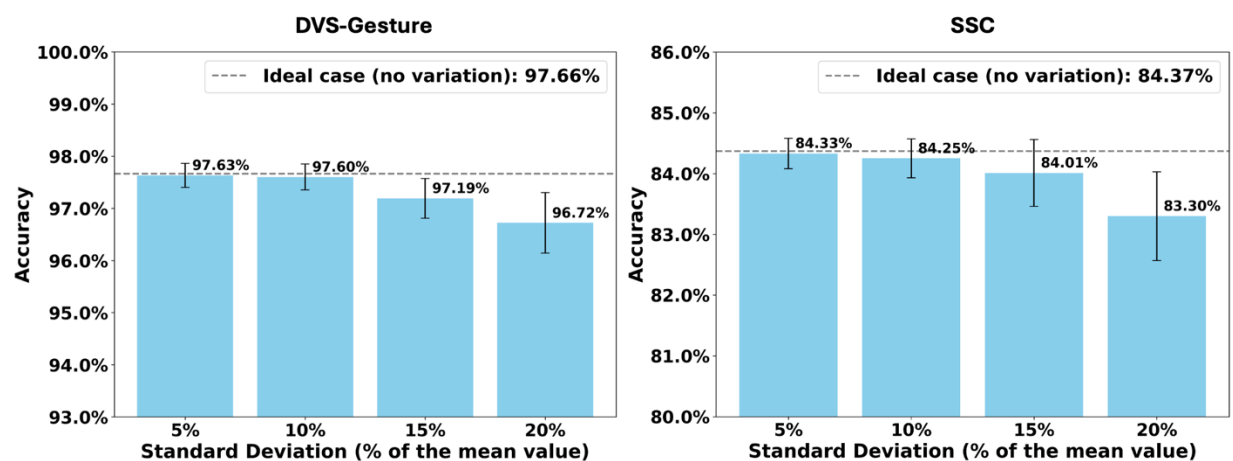


Fig S5. Model accuracy as a function of device variations in the decay constant λ . The decay constant λ of the STM memristor devices are modeled to follow a Gaussian distribution centered on the target mean value, and the standard deviation of the distribution is increased from 5% to 20% of the mean. Error bars represent results from 40 independent test runs at each variation level. The dotted line represents the ideal baseline accuracy without variation from the quantized-weight model.

Physical device fabrication is subject to inherent process variability, meaning the actual decay constant λ of a fabricated STM memristor device will likely deviate from its intended target value. To analyze the model's sensitivity to device-to-device variations, we modeled the device decay constant λ as following a Gaussian distribution centered on the layer's target mean λ . The standard deviation of the distribution was increased from 5% to 20% of the target mean λ . The model's accuracy was then evaluated by randomly

assigning λ values to the STM memristor devices following the distribution, and repeated 40 times for each distribution to obtain statistically significant results.

The results, plotted in Fig. S5, show that the model maintains high accuracy even with substantial λ variations up to 10% of the target mean λ . This result suggests that the proposed network can tolerate inherent variability of physical devices, confirming the feasibility of our hardware co-design approach. '

We also added relevant discussions to the “Hardware System Performance Analysis” Section in p14:

“Since the STM memristors need to implement the pre-trained λ , we also analyzed the effects of variations in the device decay rate on model performance (Supplementary Information S5). These results show that the models exhibit negligible accuracy degradation with variations up to 10% of the mean, without any re-training or variation-aware training techniques.”

Reviewer #2:

The authors have fully addressed all of my concerns. I expect this paper will help advance the field.

Response:

Thanks for your positive comments.

Reviewer #3:

The previous comments have been addressed by the authors.

Response:

Thank you for your positive comments.

1. However, I am wondering why the authors claim that the transformer is not compatible with CIM, given that numerous CIM studies have been proposed for transformer-based architectures.

Response:

Thank you for raising this question which helps us further clarify the challenges for implementing Transformer models in CIM. In the revised manuscript, we specifically

explained the mismatch between the attention mechanism in Transformers and the CIM system:

In p2:

‘In Transformers, the attention mechanism involves the multiplications of the Query, Key, and Value matrices, all of which depend on the input so the attention operation is not friendly to stationary CIM arrays which assume one of the matrix (i.e. weights) is static. Consequently, practical implementations of Transformers adopt hybrid designs: they leverage stationary CIM only for VMM involving static weights (e.g., in Fully Connected Layers), but delegate the input-dependent attention score computations or the KV cache management to separate digital circuitry²⁷.’

27. Büchel, J. et al. Efficient scaling of large language models with mixture of experts and 3D analog in-memory computing. *Nat Comput Sci* 1–14 (2025) doi:10.1038/s43588-024-00753-x.

2. Regarding the decay parameter (λ), the authors mention that a fixed value is used. Could the authors discuss whether this parameter may vary due to temperature fluctuations or temporal decay, and how such variations might affect the application performance or computational accuracy?

Response:

We thank the reviewer for this valuable comment. Since in our implementation λ is determined during offline training and then implemented through physical STM memristor devices, deviations from the target value can potentially affect the physical system's performance.

To address the concern about the variation in the STM memristor devices, we conducted a new simulation study to evaluate the model's robustness against variations in λ . This result is included in the revised manuscript as Supplementary Information S5. Copied below:

‘S5. Model Performance against Decay Constant (λ) Variations.

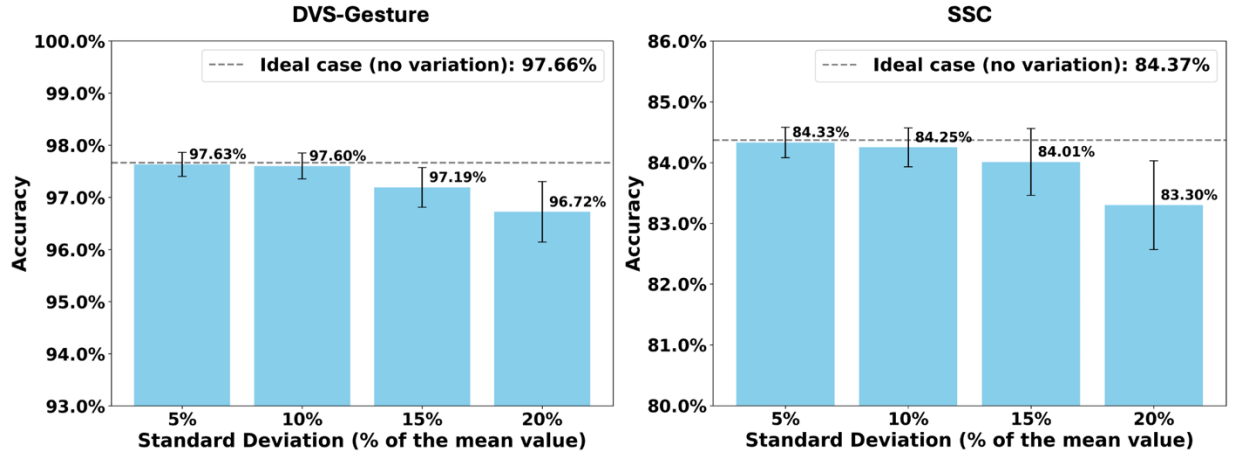


Fig S5. Model accuracy as a function of device variations in the decay constant λ . The decay constant λ of the STM memristor devices are modeled to follow a Gaussian distribution centered on the target mean value, and the standard deviation of the distribution is increased from 5% to 20% of the mean. Error bars represent results from 40 independent test runs at each variation level. The dotted line represents the ideal baseline accuracy without variation from the quantized-weight model.

Physical device fabrication is subject to inherent process variability, meaning the actual decay constant λ of a fabricated STM memristor device will likely deviate from its intended target value. To analyze the model's sensitivity to device-to-device variations, we modeled the device decay constant λ as following a Gaussian distribution centered on the layer's target mean λ . The standard deviation of the distribution was increased from 5% to 20% of the target mean λ . The model's accuracy was then evaluated by randomly assigning λ values to the STM memristor devices following the distribution, and repeated 40 times for each distribution to obtain statistically significant results.

The results, plotted in Fig. S5, show that the model maintains high accuracy even with substantial λ variations up to 10% of the target mean λ . This result suggests that the proposed network can tolerate inherent variability of physical devices, confirming the feasibility of our hardware co-design approach. '

We also added relevant discussions to the "Hardware System Performance Analysis" Section in p14:

"Since the STM memristors need to implement the pre-trained λ , we also analyzed the effects of variations in the device decay rate on model performance (Supplementary Information S5). These results show that the models exhibit negligible accuracy degradation with variations up to 10% of the mean, without any re-training or variation-aware training techniques."

The reviewer also mentioned the effects of temperature, which can indeed also affect the memristor behavior (and affect analog computing devices and circuits in general). Unlike device variations that are random, temperature effects cause systematic errors and can potentially be modeled and compensated. In the revised manuscript, we acknowledged potential temperature effects and proposed how these systematic effects can potentially be compensated.

In p16:

‘... recent device-level research demonstrates, the time constant of the SN can be actively modulated in-situ by an external control signal, such as a bias voltage⁶⁶. Such approaches make it possible to dynamically adjust the state decay rates on-the-fly and effectively implement modern selective SSMs in the proposed architecture. Additionally, systematical shifts, such as temperature effects on the device’s decay constant, can potentially be compensated by techniques such as bias voltage control, enhancing the system’s real-world robustness.’

66. Li, Y. et al. Homogeneous Memristors with Tunable Decay Dynamics for Self-Adaptive Reservoir Computing. *ACS Nano* **19**, 28186–28196 (2025).